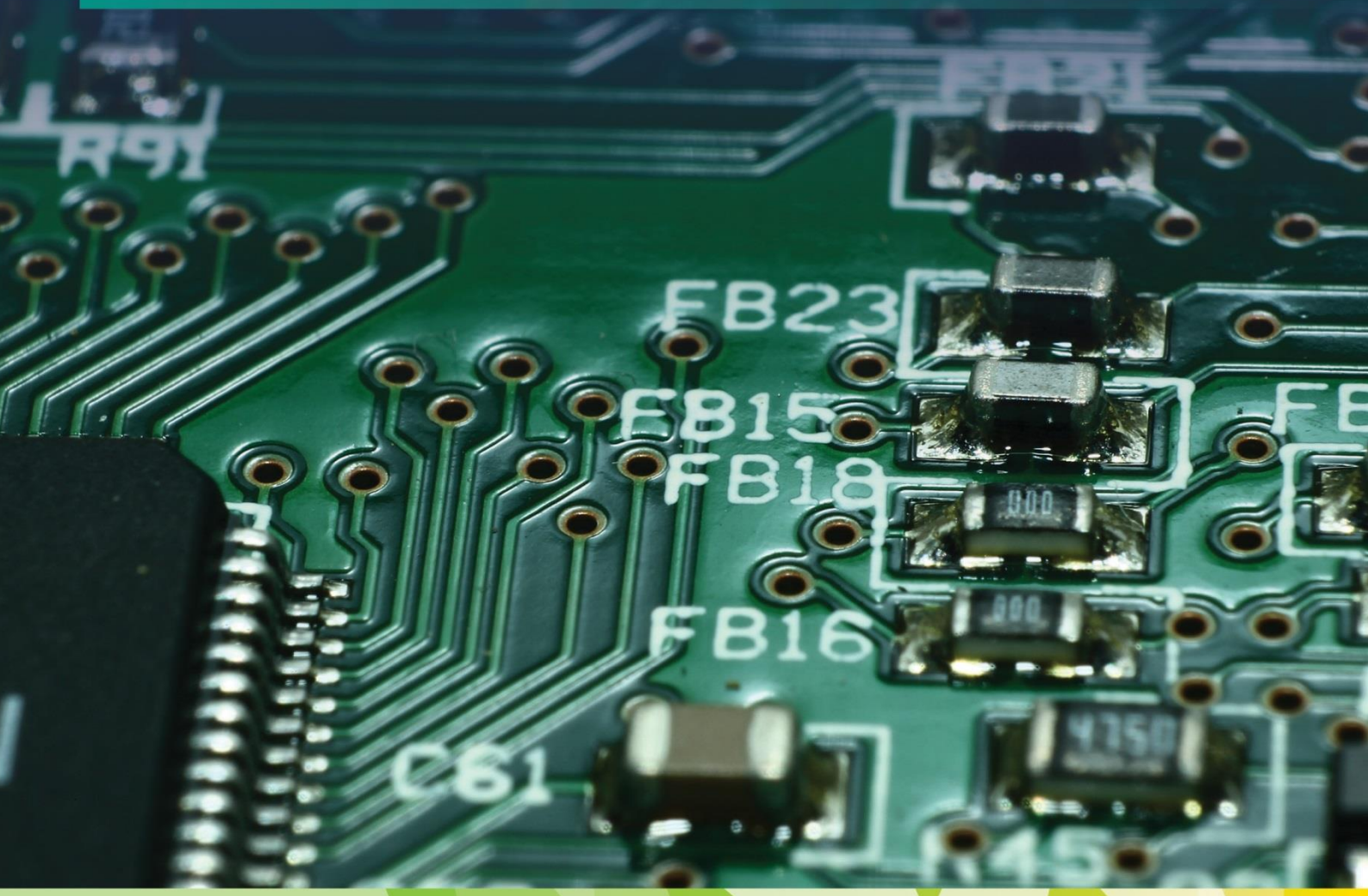
NSW GOVERNMENT | Department of Primary Industries

Teacher guide

Code and build an aquaponics control system



www.dpi.nsw.gov.au

connect • grow • learn
DPI in SCHOOLS

Supporting document

NSW DPI Schools Program

Authors: Meg Dunford (Project Officer School programs, NSW DPI Orange) and Charlene Chamberlain (Digital Laboratory Technician, Climate Branch, NSW DPI, Orange).

Editors and Advisors: Michelle Fifield (Education Officer Schools, NSW DPI Orange) and Jo Hathway (Project Officer School programs, NSW DPI Tocal College).
Design: Eddy Archer (Communications Officer, NSW DPI Orange).

Cover image Circuit board, Source Pixabay.

Contents

Information for teachers

Syllabus context

The Yabby unit Design project 1 is mapped to outcomes from *Agriculture and Digital Technologies* context of the NSW Technology Mandatory (2017) syllabus. It integrates content from agriculture technologies, and digital technologies to enable delivery.

Agriculture technologies (food and fibre production) focus on the investigation of managed environments, such as farms and plantations. Students learn about the processes of food and fibre production and investigate the innovation and sustainable supply of agriculturally produced raw materials. Students develop deep knowledge and understanding about managed systems that produce food and fibre through designing and producing solutions.

The *Digital technologies* context encourages students to develop an empowered attitude towards digital technologies, use abstractions to represent and deconstruct real-world problems, and implement and evaluate digital solutions. Students have the opportunity to become innovative creators of digital technologies in addition to effective users of digital systems and critical consumers of the information they convey. Source: NESA, 2017. Technology Mandatory Syllabus.

Resource description

This document is a teacher guide showing a possible build for digital control and automation of elements in aquaponics system for the Yabby unit Folio 1. The build and coding used is complex giving possible options for automated control technologies that could be applied in the Yabby unit, or any other aquaponics, horticultural, cropping, pastoral or animal production systems which you may have at your school.

This resource has been developed to show the step- by- step build guide, code, software use, program code, equipment and costing of the system the DPI Schools Program and DPI Climate Team built.

Equipment used and costs are dated to 2018. The equipment, cost and technology described in this document must be viewed as an example. There are many variations in electronic equipment and how this design could be built. Our build used equipment specifically selected to meet a budget achievable to school settings.

Arduino software, microcontrollers and compatible electronic components were used, as they align with the NSW Department of Education's Crack the Code Unit resource which was rolled out in 2018 and provided teacher training, resources and electronic kits to NSW Department of Education schools around the state.

Before undertaking this build it is advised teachers have prior understanding and basic skills using digital technologies. You will require some background and fluency in general purpose programming language and electronics skills in order to complete this project.

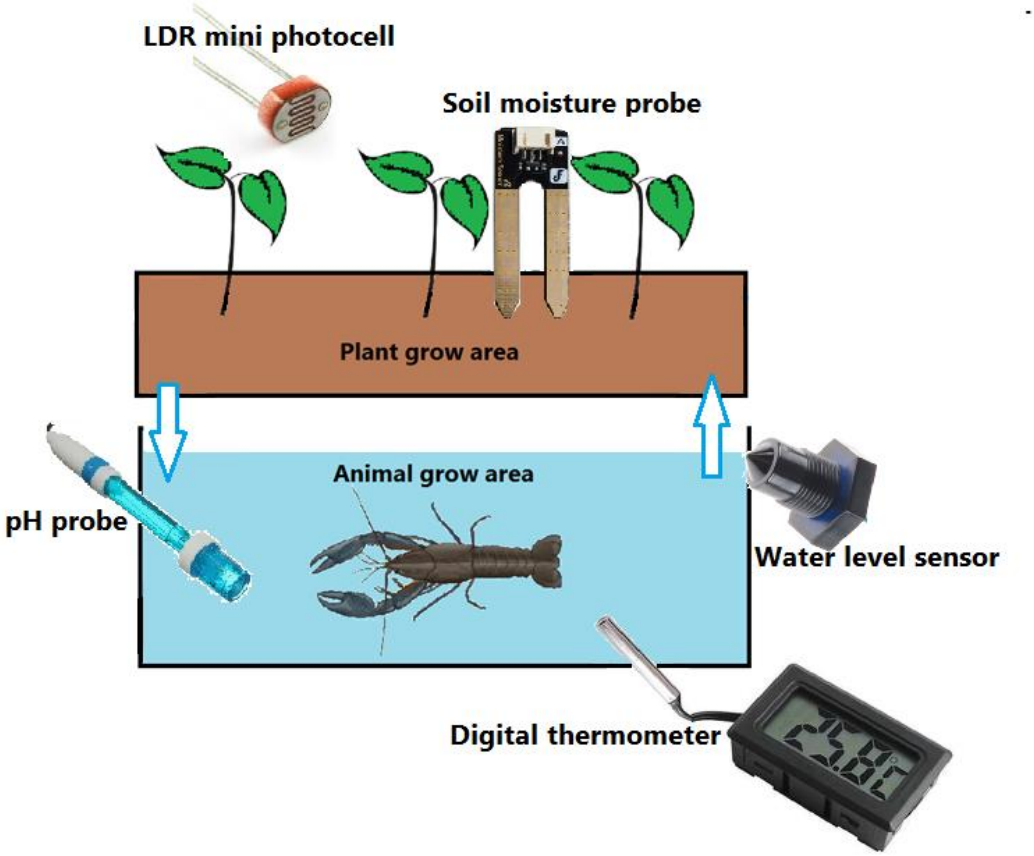
If you have no prior background knowledge, the Appendix readings at the end of this resource give some tips, however it is theoretical reading. Development of skills occurs through practical application, so it is advised for teachers to access equipment to build and develop digital systems.

There are also many free online courses, tutorials and support available on the internet. Good sites for Arduino technologies, include the Arduino homepage https://www.arduino.cc/ free learning tutorials, and Core Electronics for more free courses https://core-electronics.com.au/tutorials/arduino-workshop-for-beginners.html.

Design and code an aquaponics system - teachers guide

What you are building

By following this guide and using the provided code and suggested components you can build a digitally automated system with the following sensors.



This build is suggested to be used to digitally automate and control an aquaponics system; however the digital system and components selected could be used for a range of applications in disciplines agriculture, science, horticulture, aquaculture, to capture data.

Components and functionality

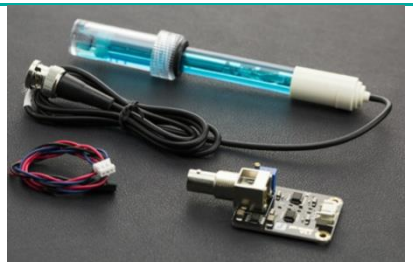
Component	Description	Functionality in this build	Other uses
LDR mini photocell	An LDR (light dependent resistor), is a component that has a (variable) resistance that changes with the light intensity that falls upon it. This allows them to be used in light sensing circuits. LDR's measure light intensity in units LUX.	In this system the LDR component determines the light intensity in LUX (i.e. light or dark) and is programmed to turn on a red LED globe when the LUX level is low (dark).	LDR's are commonly used to automatically turn on a light at a certain light level. An example of this could be a street light, night light or a grow light to promote plant growth.
Soil moisture probe	Soil moisture sensors are designed to estimate soil volumetric water content based on the dielectric constant (soil bulk permittivity) of the soil. The dielectric constant can be thought of as the soil's ability to transmit electricity. The dielectric constant of soil increases as the water content of the soil increases.	The soil moisture probe is programmed to turn on an indicator blue LED globe when the soil moisture level is lower than 60%.	Soil moisture probes have many applications including being used for irrigation scheduling, connecting to equipment that can send messages and alerts or automatically start pump systems.
Water level sensor (liquid level sensor)	Level sensors are used to monitor and regulate levels of a particular free-flowing substance (liquid) within a contained space. There are many different types of liquid level sensors; the type used in this build is a photoelectric sensor. It works by determining the proximity of water by transmitting an infrared beam towards the liquid. The beam refracts off the liquid and the photo sensor measures how far the beam travelled, thus determining proximity.	The photoelectric liquid level sensor in this build could be used to determine the water level in the water sump or animal grow area in an aquaponics system. The sensor is coded to turn on an indicator yellow LED to alert when the water level has dropped below the level of the sensor. The yellow LED alerts that the tank needs to be filled.	Level sensors are widely used industrially. Examples include liquid level sensors to monitor fuel and oil in cars and machinery; sensors and in industrial storage tanks. In agriculture they are frequently used in silos, water tanks, dams and water troughs. The sensor could also be used to log data for example to link to humidity, evaporation or climatic changes.
Digital thermometer	Digital thermometers do not contain mercury, but contain a thermistor chip inside the tip which is used to measure the temperature. Digital thermometers rely on the principle that the electrical resistance of metal changes with temperature. The resistance increases as the temperature rises and lowers as it	The digital thermometer in this build could be used in either the plant or animal grow areas (The component we used is waterproof to allow it to be fully immersed in water). The sensor is coded to turn on an indicator white LED to alert	Digital thermometers have a massive range of uses. In agriculture they could be used to log data for example to link to humidity, evaporation or climatic changes or directly

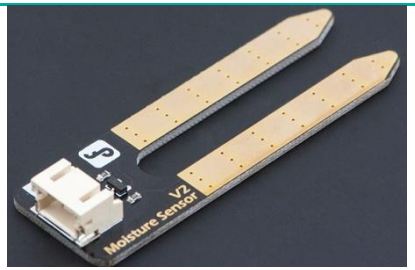
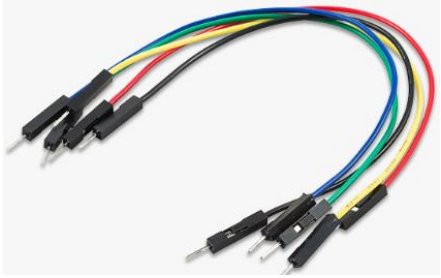
	cools. The digital thermometer works by passing a voltage through an electrode and measuring the current, which changes with temperature.	when the temperature exceeds 25 degrees Celsius.	in plant and animal growth and health management and monitoring.
pH probe	pH is a measurable parameter between the values of 0 and 14. Solutions with a pH<7 are acidic, whereas those with a pH>7 are alkaline. A pH meter is an electronic device that measures the changes in the activity of hydrogen ions in solution. pH meters are comprised of a special measuring probe (a glass electrode or, an ion-selective field-effect transistor (ISFET), attached to an electronic meter that displays the decimal pH reading. The pH meter must be calibrated prior to use against buffer solutions of known hydrogen ion activity for accuracy.	The pH probe in this build is a glass electrode probe and could be used in either the plant or animal grow areas to measure the pH of the soil solution or animal water, which could affect plant and animal growth. The sensor is coded to turn on two green LED's to alert when the pH is outside the range pH6.5-7.5. If the pH is less than 6.5 one green LED wil light up, if the pH is greater than 7.5, the two green LED's will light up.	pH probes have a massive range of scientific field or laboratory applications. They can be used to measure the pH of any water based solution, and log pH data for, water, soil, plant and animal management and monitoring.

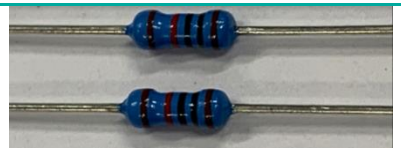
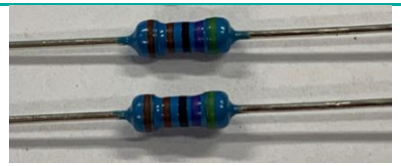
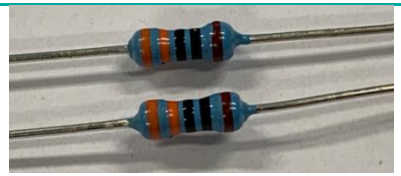
Equipment list

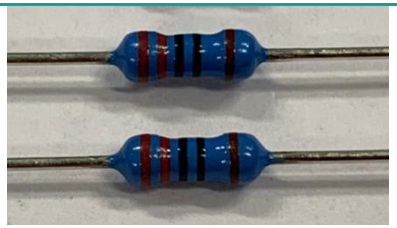
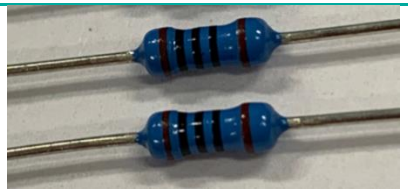
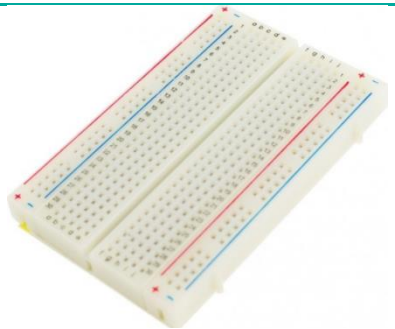
This list should be used as a guide. Use your networks, suppliers and resources available to you to make purchase and build decisions.

Part	Image	Description	Quantity	Price at (Dec 2018)
Arduino IDE software		The freely available open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. Download the latest version at: https://www.arduino.cc/en/main/software	Arduino IDE software	• Free
Arduino Uno R3 Microcontroller or similar compatible microcontroller **(Many cheap versions available- showing **DFRduino UNO R3 - Arduino Compatible)**		The Arduino UNO is a microcontroller board. It plugs into a computer with open source IDE installed software. Controls are loaded onto the board giving it functionality. Extensions are added to the board to give it physical functionality.	1	• Arduino Uno R3 Microcontroller $7.00-$30.00

Waterproof DS18B20 Digital Temperature Sensor for Arduino		This waterproof digital temperature sensor chosen is simple to wire, whilst measuring temperature in a vast range, with high accuracy. Usable with 3.0V-5.5V power/data and ±0.5°C Accuracy from -10°C to +85°C.	1	$6.90
Gravity: Photoelectric Water / Liquid Level Sensor For Arduino		This is a photoelectric liquid level sensor that operates using optical principles. It provides a single point level detection. This product has no mechanical parts which reduced amount of calibration required. Liquid level detection accuracy ±0.5mm.	1	$5.50
Gravity: Analogue pH sensor kit for Arduino		This analogue pH meter, is specially designed for Arduino controllers and has convenient and practical "Gravity" connection, which allows it to instantly connect between the probe and Arduino to get pH measurements at ± 0.1pH (25 ℃).	1	$35.00

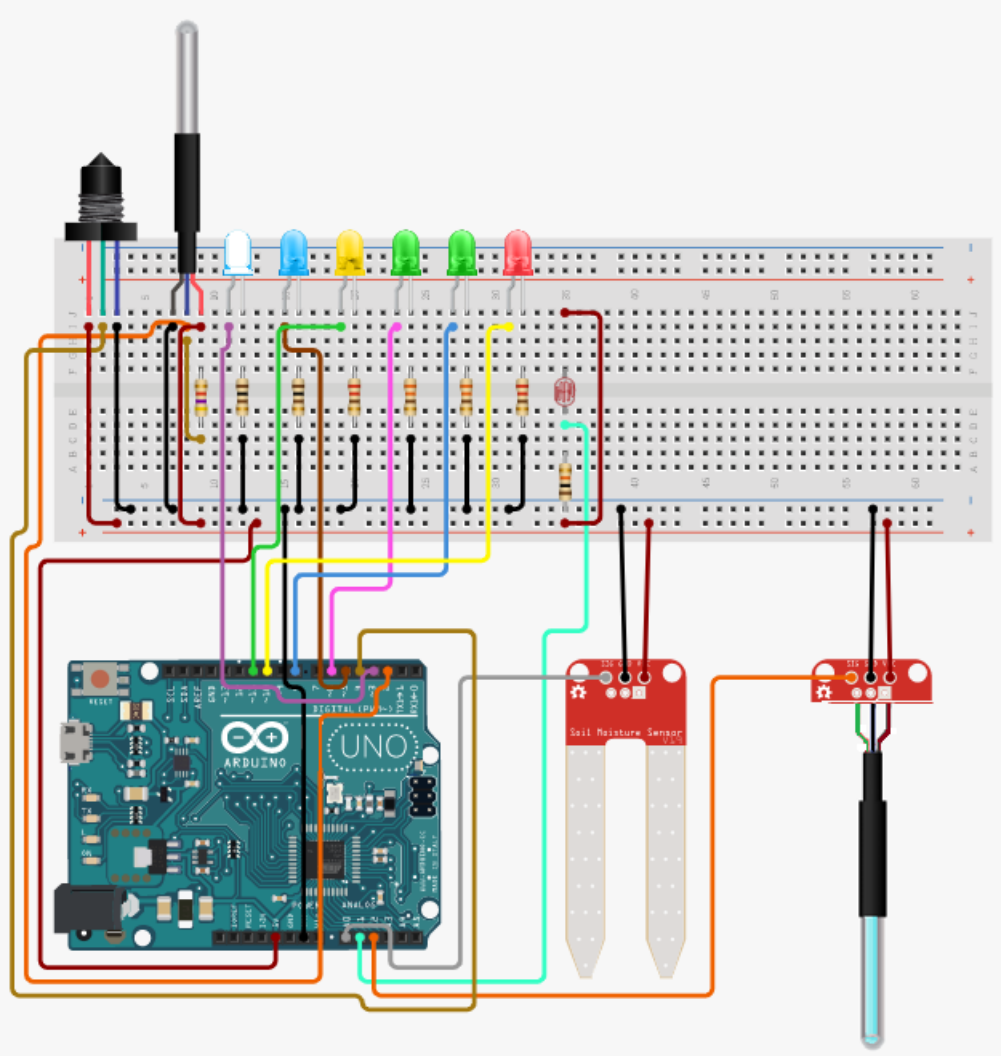
Gravity: Analog Soil Moisture Sensor For Arduino		This soil moisture Arduino compatible sensor, uses two probes to pas and measure electric current in the soil. It reads the resistance to determine moisture level. More water increases soil conductivity, less water reduces conductivity. This product easily plugs into the Gravity IO expansion shield.	1	$2.70
Light dependent resistor-LDR (mini photocell) sensor		This is a small light sensor that can measure changes in ambient light including loss of sunlight or loss of artificial light (light bulb). Returned values: from 0 (no light) to 1023 lumens (maximum light). Sample applications include detecting light has been turned out to control an output function.	1	$0.60
Jumper wires M/M		Jumper wires are essential to connect components within the circuit. These easy to use 'plug in' varieties have been used for this build. A range of male: male; male: female and female: female plug in types are required if using these wire types.	3 (10 pack)	$1.75

Jumper Wires F/M		Jumper wires are essential to connect components within the circuit. These easy to use 'plug in' varieties have been used for this build. A range of male: male; male: female and female: female plug in types are required if using these wire types.	1 (10 Pack)	$1.75
10K Ohm resistor **5 band - brown, black, black, red, brown**		A resistor is a passive two- terminal electrical component that implements electrical resistance as a circuit element. Resistors are used to reduce current flow, adjust signal lengths, and divide voltages among other uses. The number, thickness and colour of the bands on the resistor indicate its resistance value. Resistors have 6, 5 or 4 bands for universal identification.	1	$0.013
4.7K Ohm resistor **5 band - yellow, violet, black, brown, brown**		A resistor is a passive two- terminal electrical component that implements electrical resistance as a circuit element. Resistors are used to reduce current flow, adjust signal lengths, and divide voltages among other uses. The number, thickness and colour of the bands on the resistor indicate its resistance value. Resistors have 6, 5 or 4 bands for universal identification.	1	$0.013
330 Ohm resistor **5 band - orange, orange, black, black, brown**		A resistor is a passive two- terminal electrical component that implements electrical resistance as a circuit element. Resistors are used to reduce current flow, adjust signal lengths, and divide voltages among other uses. The number, thickness and colour of the bands on the	1	$0.013

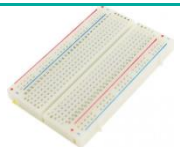
		resistor indicate its resistance value. Resistors have 6, 5 or 4 bands for universal identification.		
220 Ohm resistor **5 band - red, red, black, black, brown**		A resistor is a passive two- terminal electrical component that implements electrical resistance as a circuit element. Resistors are used to reduce current flow, adjust signal lengths, and divide voltages among other uses. The number, thickness and colour of the bands on the resistor indicate its resistance value. Resistors have 6, 5 or 4 bands for universal identification.	2	$0.013
100 Ohm resistor **5 band - brown, black, black, black, brown**		A resistor is a passive two- terminal electrical component that implements electrical resistance as a circuit element. Resistors are used to reduce current flow, adjust signal lengths, and divide voltages among other uses. The number, thickness and colour of the bands on the resistor indicate its resistance value. Resistors have 6, 5 or 4 bands for universal identification.	1	$0.013
Basic LED 5mm 5 × colours		Small, basic LED's in a range of colours. Intensity of colour may be controlled at an approximate range 0-255.	6	$0.20 each
Half size breadboard		Half size solderless breadboard. A breadboard is a solderless device for temporary prototype with electronics and test circuit designs. Most electronic components in electronic circuits can be interconnected by inserting their leads or terminals into the holes and then making connections through wires where appropriate. Breadboards have strips of metal underneath the board that connect to the holes on the top of the board. Price dependant on size.	1	$7.85
			Approximate total cost of full build as at December, 2018	**$73.00**

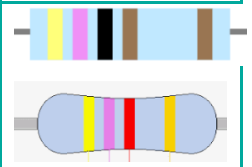
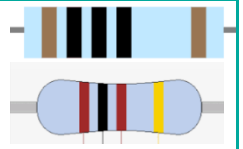
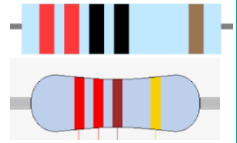
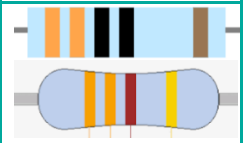
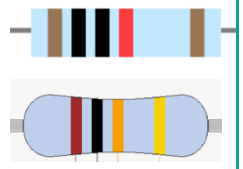
Step–by–step wiring guide

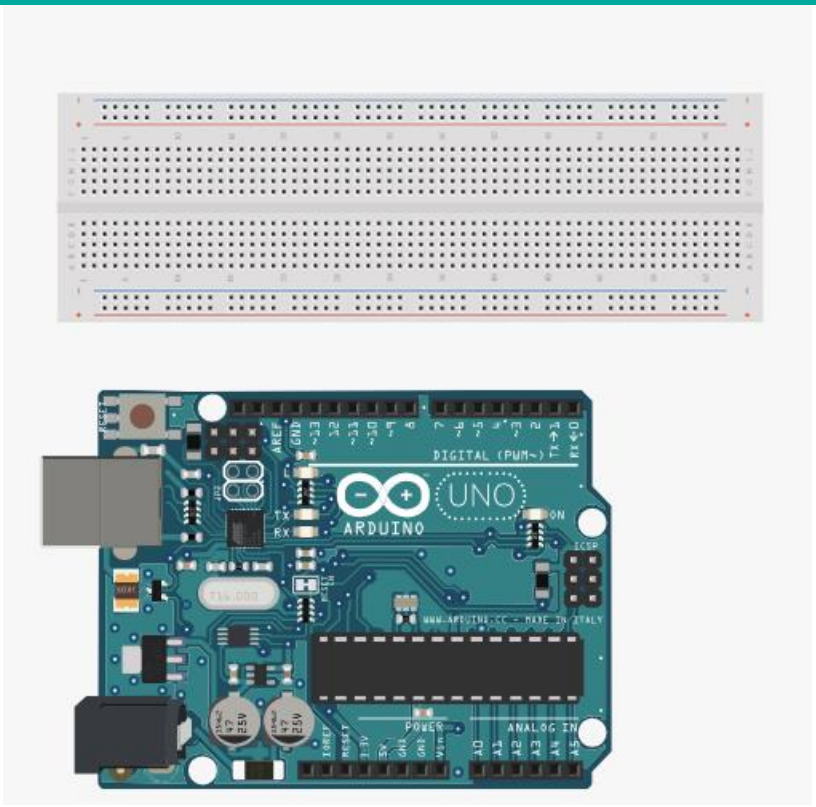
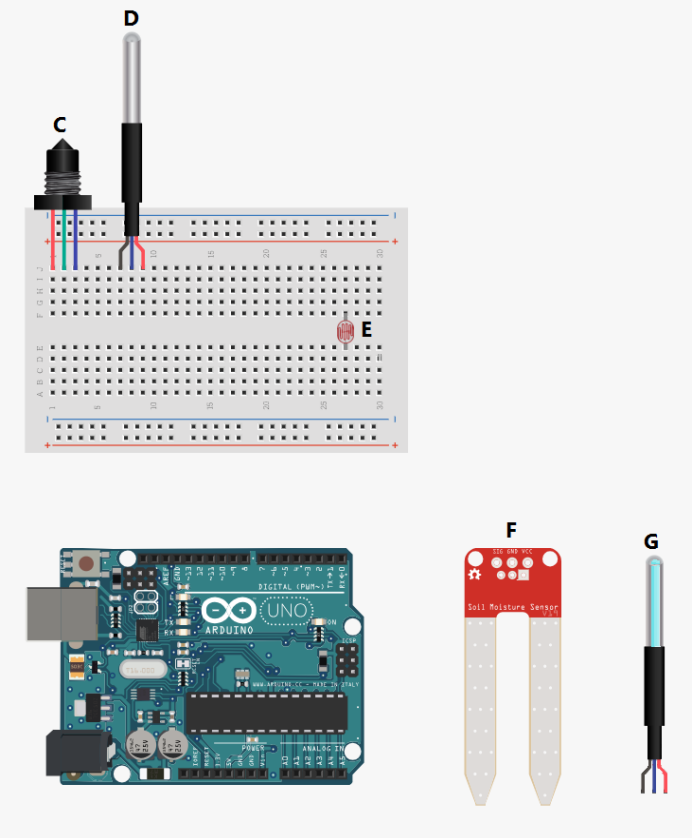
All illustrations and wiring guide in this section have been provided using "circuito.io" a free online circuit builder.

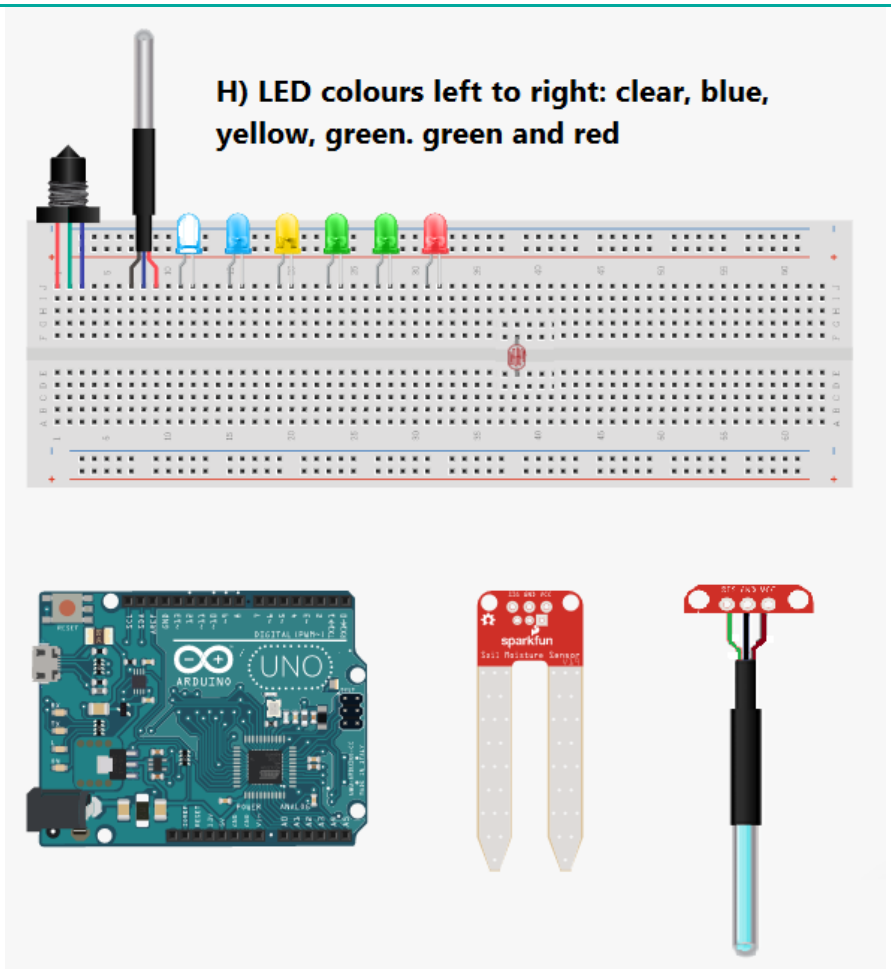


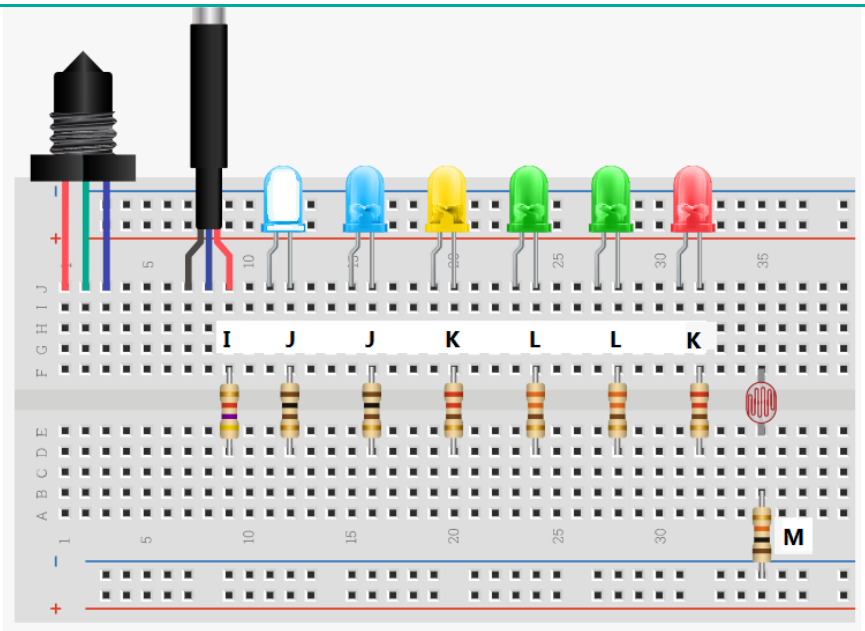
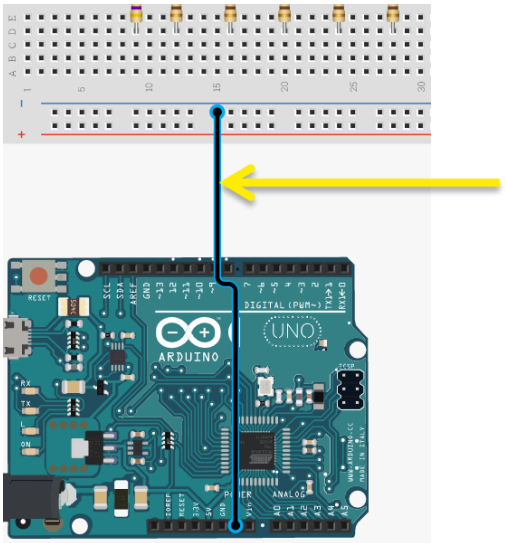
Source: circuito.io

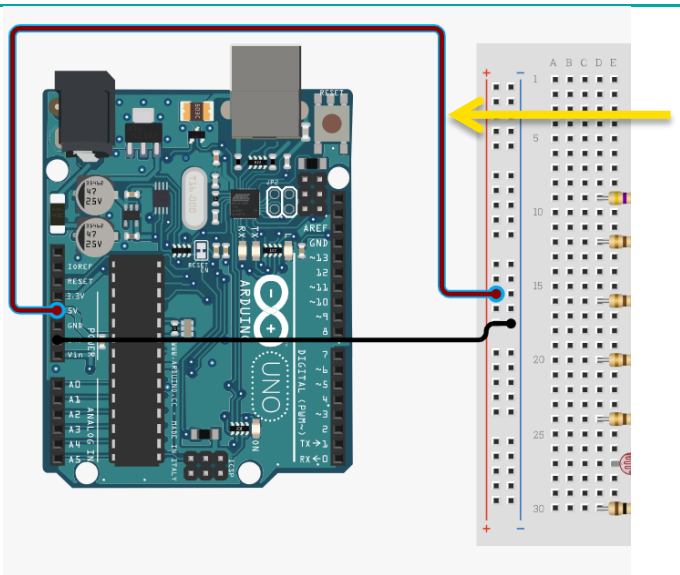
Part number	Part	Image
A	Microcontroller	
B	Breadboard	
C	Photoelectric Water / Liquid Level Sensor	
D	Digital Temperature Sensor for Arduino	
E	LDR (mini photocell) sensor	
F	Analogue Soil Moisture Sensor	
G	Analogue pH sensor	

H	LED's Range of colours	
I	4.7K Ohm resistor 5 band - yellow, violet, black, brown, brown 4 band- yellow, violet, red, gold	
J	100 Ohm resistor 5 band - brown, black, black, black, brown 4 band - brown, black, brown, gold	
K	220 Ohm resistor 5 band - red, red, black, black, brown 4 band - red, red, brown, gold	
L	330 Ohm resistor 5 band - orange, orange, black, black, brown 4 band – orange, orange, brown, gold	
M	10K Ohm resistor 5 band - brown, black, black, red, brown 4 band – brown, black, orange, gold	
N	Jumper wires	
O	Arduino software IDE. Free download at https://www.arduino.cc/en/Main/Software	

Connecting all inputs, outputs and components		
Step	**Description**	**Image**
1	Set up Microcontroller and breadboard (parts A and B)	
2	Place and connect all sensors (parts C-G) as seen in the diagram. Water level sensor (part C), temperature sensor (part D), LDR photocell sensor (part E), soil moisture sensor (part F) and pH sensor (part G). It doesn't matter exactly which columns you place the components but make sure you will leave enough space between all components	

	for wiring.	
3	Connect LED's as seen in the diagram. Note the 'bent leg' in the diagram of the LED indicates the longer leg of the LED. The longer leg is the positive anode pin and the shorter leg is the negative cathode pin. With circuitry it is important to get polarity (positive/ negative) correct when wiring or the circuit will not work. When trouble shooting a faulty circuit always check components are wired correctly according to polarity. Most components will label + or – indicated either through signing or coloured wiring (+ve is red; -ve is black).	

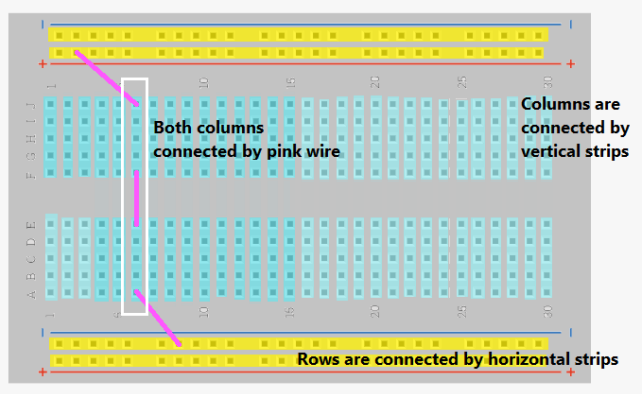
4	Connect resistors (parts I-M) as shown in the diagram. Resistors in order from left to right are 4.7K Ohm resistor (I), 100 Ohm resistor (J), 100 Ohm resistor (J), 220 Ohm resistor (K), 330 Ohm resistor (L), 330 Ohm resistor (L), 220 Ohm resistor (K), and 10K resistor (M). Note Part M is connected directly into GND.	
Connecting the Arduino UNO microcontroller. All the following connections will use jumper wires (part O). All wires in diagrams are highlighted in blue. It does not matter what colour wire you use.		
5	Connect Arduino UNO common connection Ground (GND) point to breadboard common connection point GND. A common connection point may be abbreviated as 'Bus'. For example 'Bus GND'. The term 'Bus' is a contraction of the Latin *omnibus*; meaning 'for all' i.e. terminal for all or common connection point. The term pin or point may be used interchangeably.	

6	Connect Arduino UNO 5V pin to Bus Positive terminal pin (Bus POS). This connection is setting up to provide current to the circuit.	

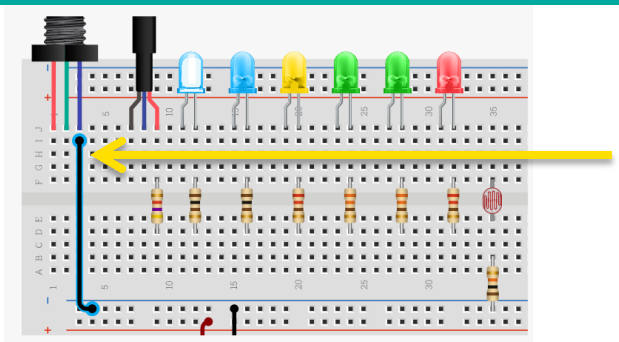
Step 6 connects current (energy) and code from the controller to the breadboard. This will enable all components to be connected to power and for control code to be applied.

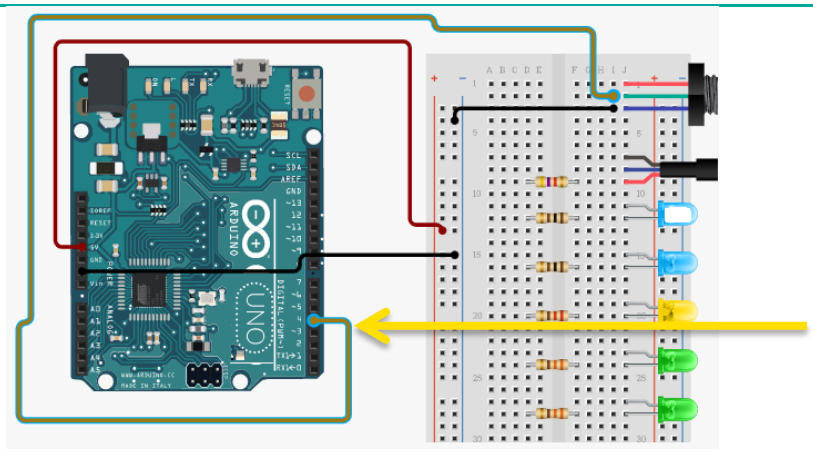
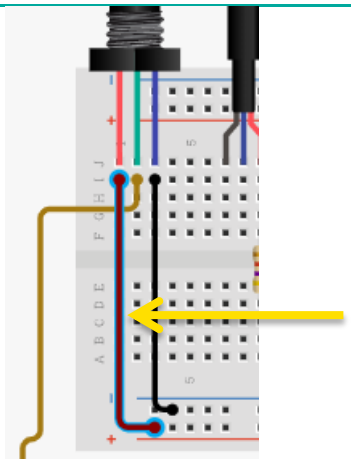
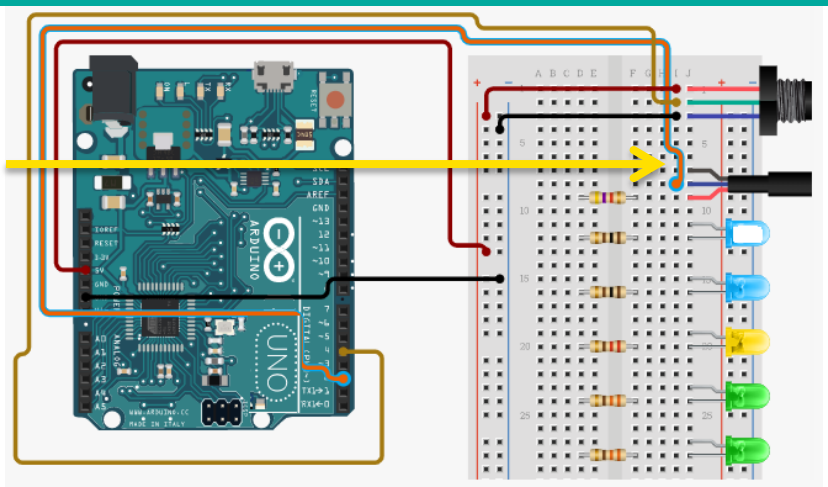
Breadboards are wired as shown. Top and bottom rows of holes are connected horizontally (yellow) and split in the middle while the remaining holes are connected vertically (blue). A set of connected holes are called a node and are connected by a metal strip beneath the breadboard.

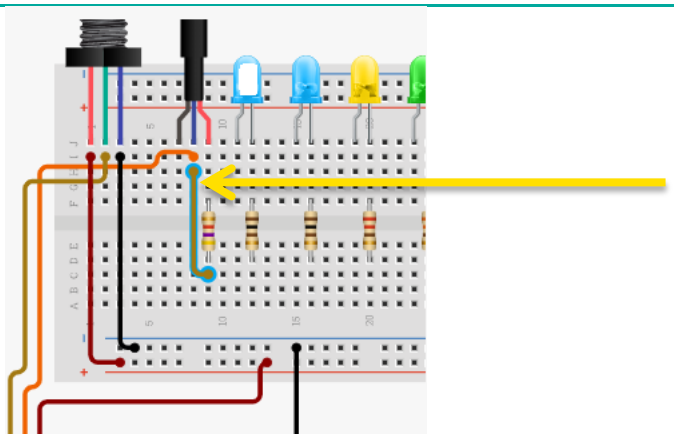
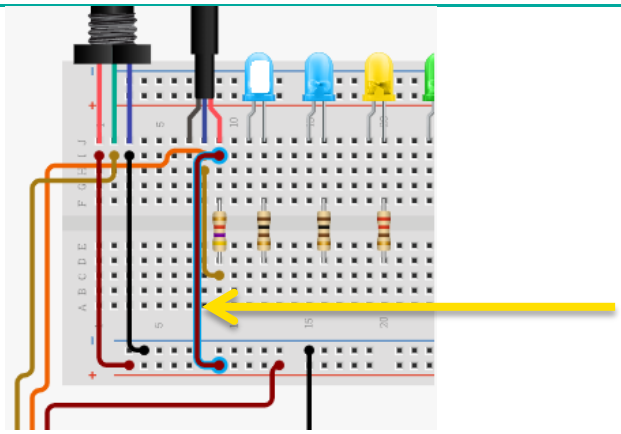
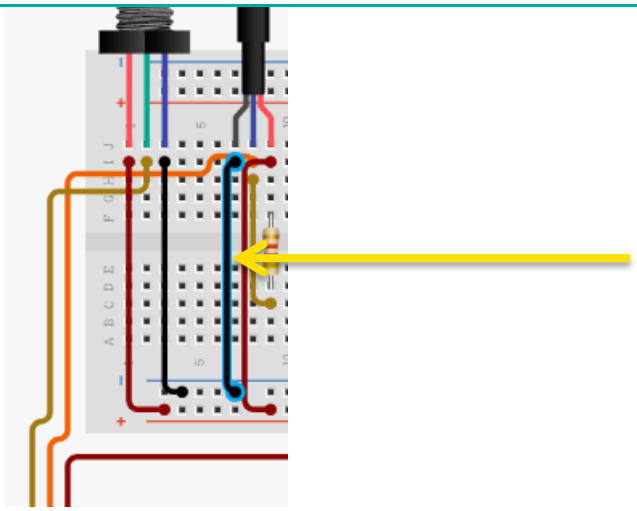
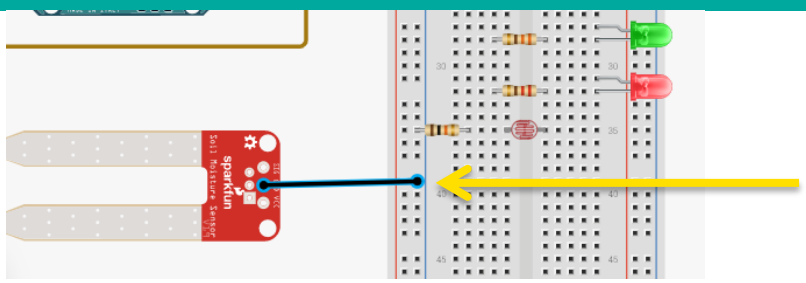
To interconnect these either connective wires or components are used (for example, see the pink wire in the diagram).

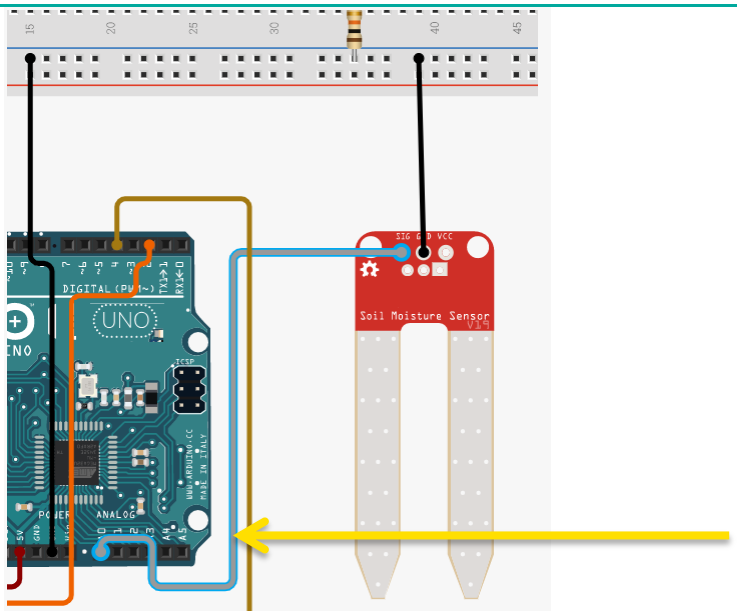
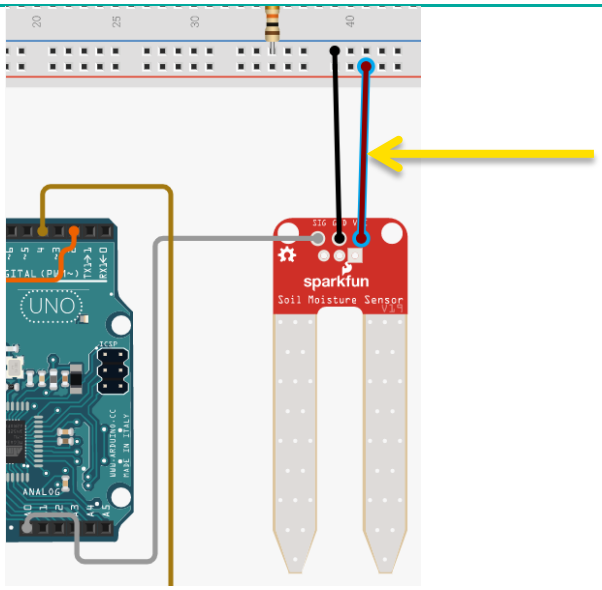
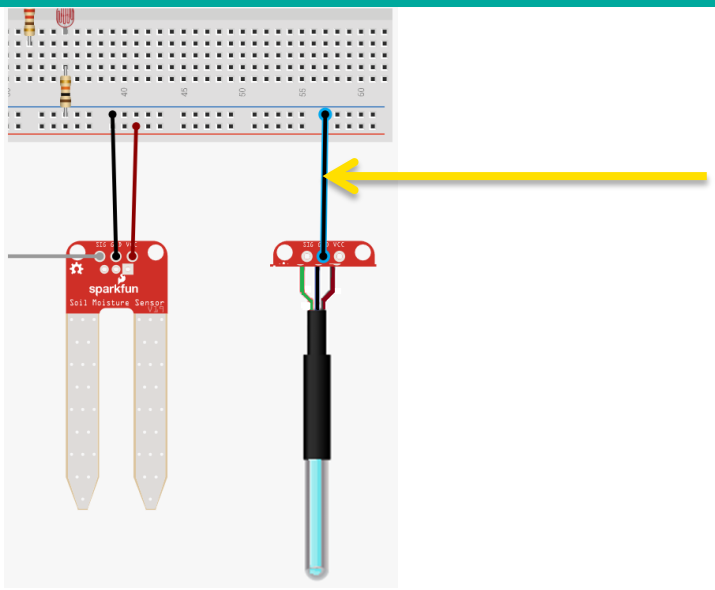


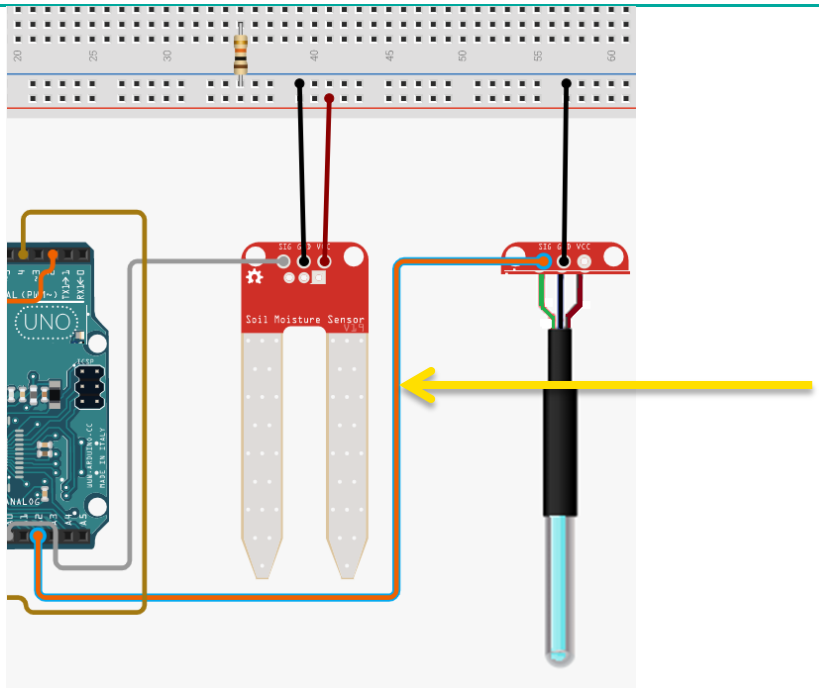
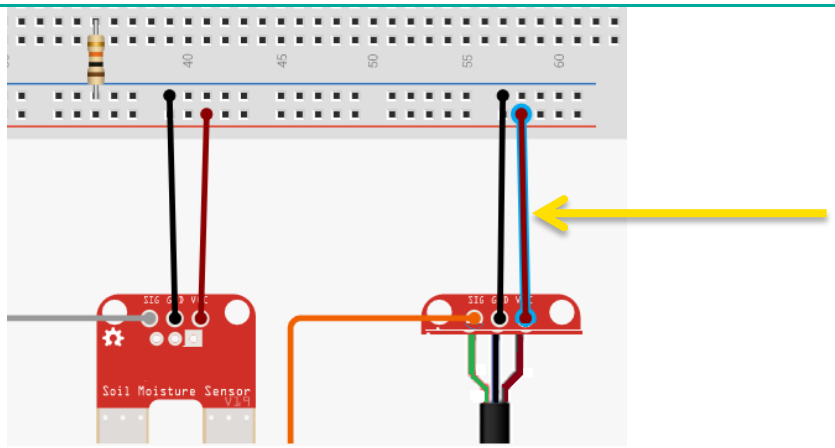
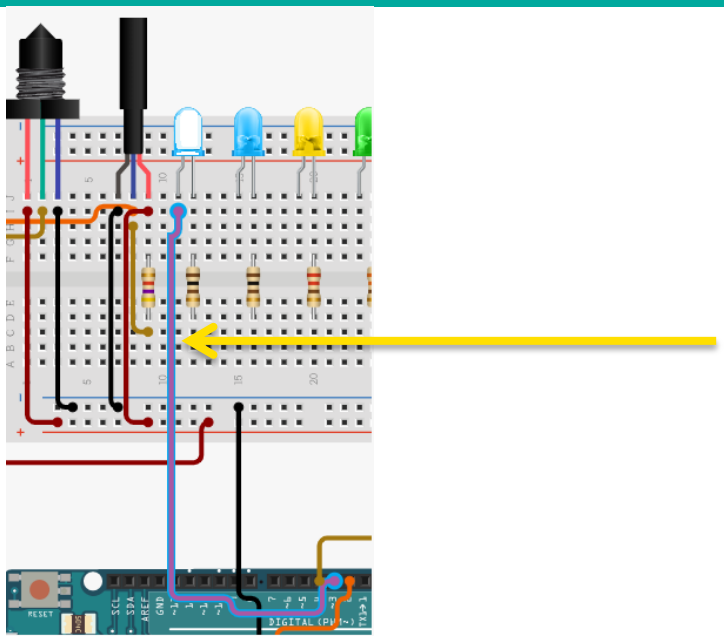
Connect water level sensor

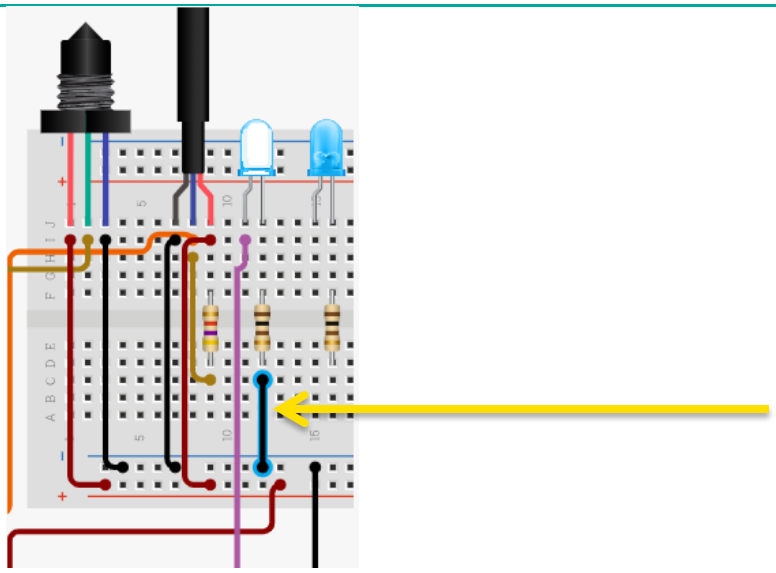
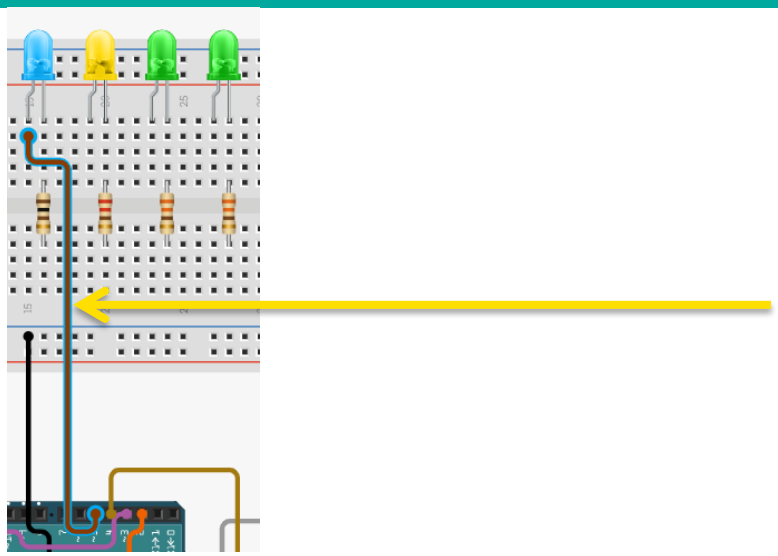
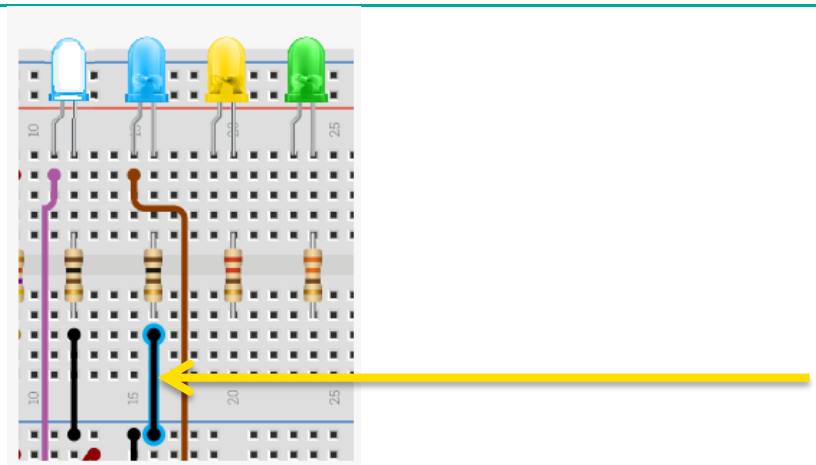
7	Connect Part C GND to Bus GND	

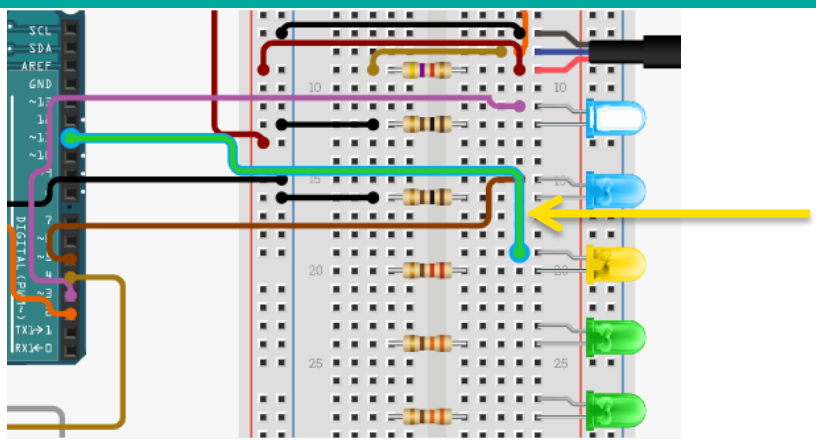
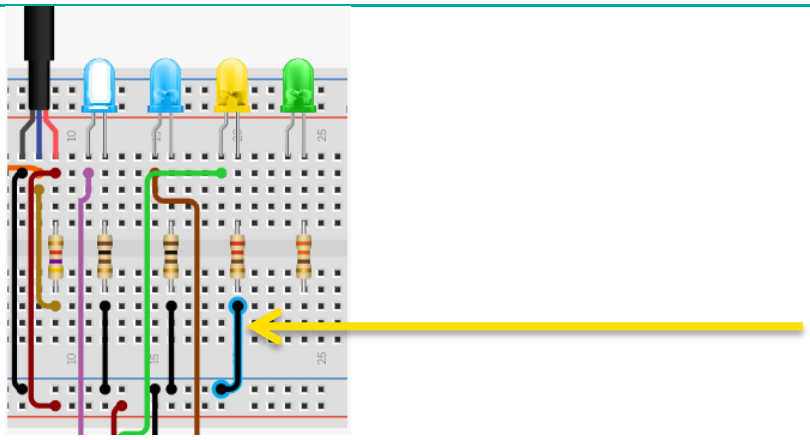
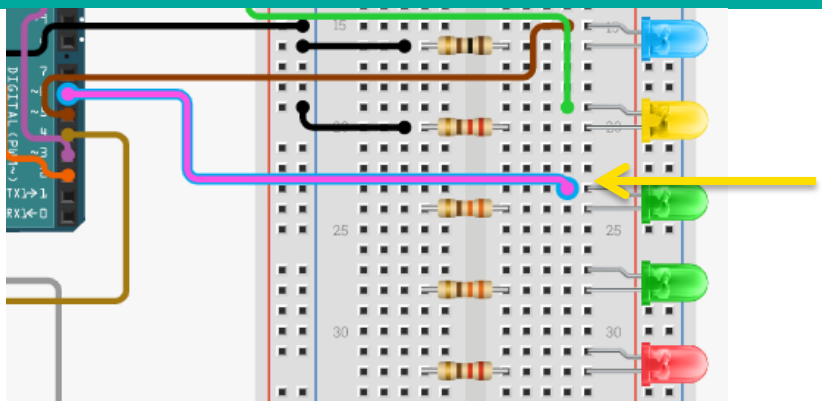
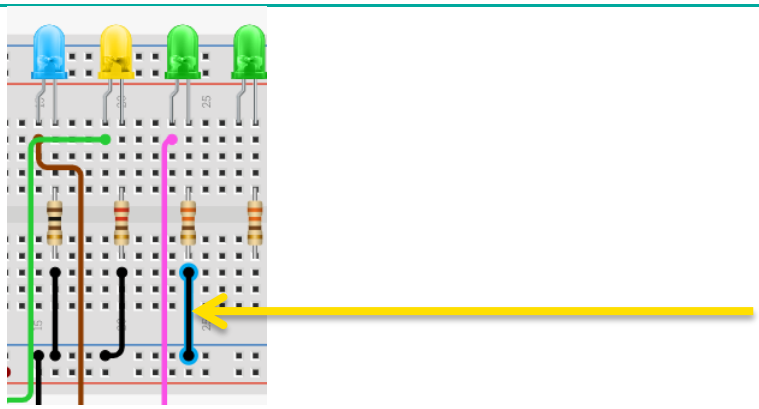
8	Connect part C signal output point (SIG) to Arduino UNO signal pin 4. When writing and manipulating the code digital pin 4 will control the water level sensor.	
9	Connect Part C's voltage common connector (VCC) point to Bus POS	
Connect digital temperature sensor		
10	Connect part C middle wire to Arduino UNO 2 digital pin. When writing and manipulating code digital pin 2 will control the digital temperature sensor.	

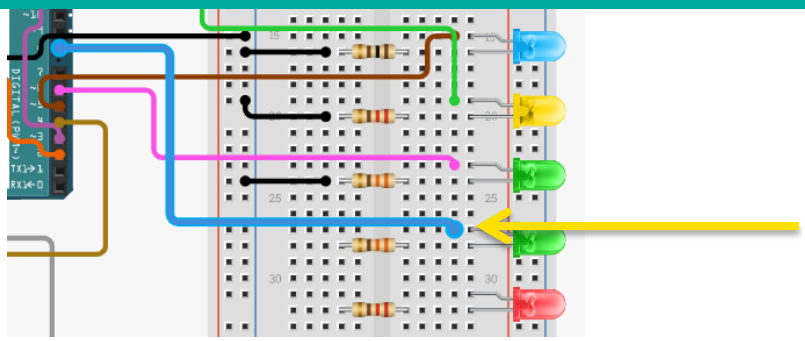
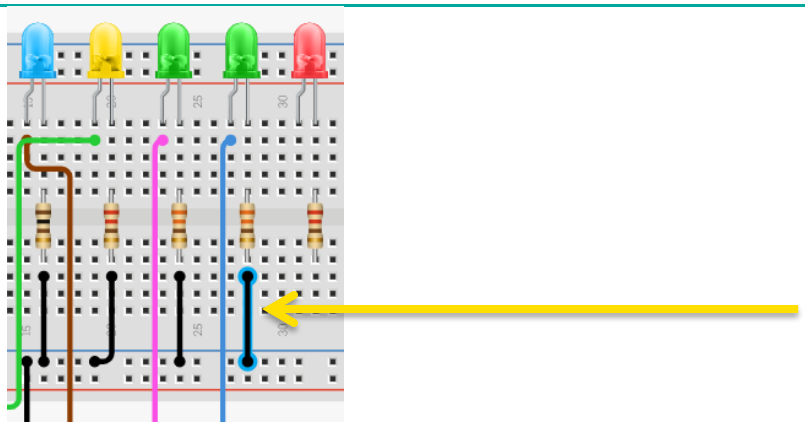
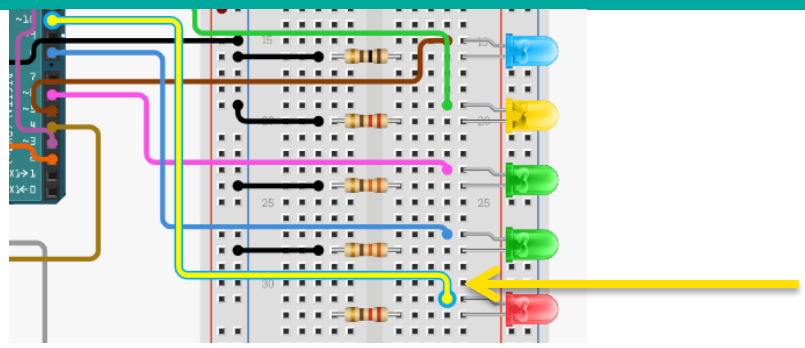
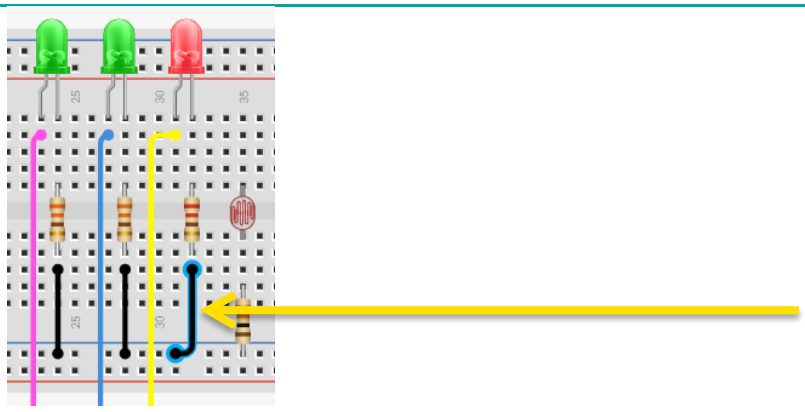
11	Connect temperature sensor to Part I (4.7K Ohm resistor) by connecting middle wire to below Part I.	
12	Connect Part I resistor and digital thermometer to power by connecting POS to Bus POS	
13	Connect digital thermometer to ground by connecting GND to Bus GND	
Connect analogue soil moisture probe		
14	Soil moisture probe (part F) GND to Bus GND	

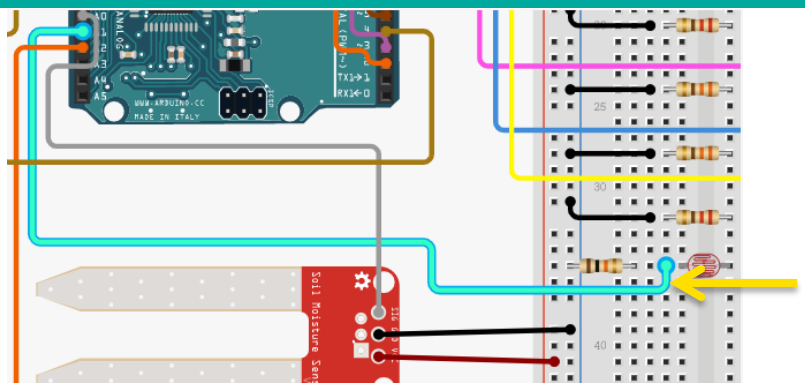
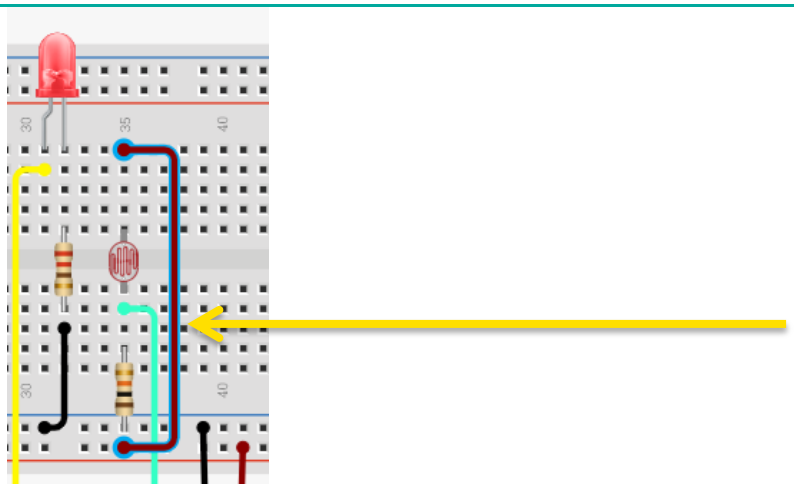
15	Part F SIG pin to Arduino UNO analogue pin A0. When writing and manipulating the code, analogue pin A0 will control the soil moisture sensor.	
16	Part F VCC pin to Bus POS	
Connect analogue pH sensor		
17	pH sensor (part G) GND pin to Bus GND	

18	Part G SIG pin to Arduino UNO analogue pin A2. When writing and manipulating the code, analogue pin A2 will control the pH sensor.	
19	Part G VCC pin to Bus POS	
Connect LED Clear		
20	Connect LED clear (Voltage In) VIN to Arduino UNO Digital pin 3. When writing and manipulating the code, digital pin 3 will control LED clear.	

21	Connect 100 Ohm resistor (Part J) to Bus GND	
Connect LED Blue		
22	Connect LED blue VIN to Arduino Uno digital pin 5 When writing and manipulating the code, digital pin 5 will control LED blue.	
23	Connect part J (100 Ohm resistor) to Bus GND	

Connect LED Yellow		
24	Connect LED yellow to Arduino UNO digital pin 11. When writing and manipulating the code, digital pin 11 will control LED yellow.	
25	Connect Part K (220 Ohm resistor) to Bus GND.	
Connect LED green		
26	Connect LED green VIN to Arduino digital pin 6. When writing and manipulating the code, digital pin 6 will control LED green.	
27	Connect part L (330 Ohm resistor) to Bus GND	

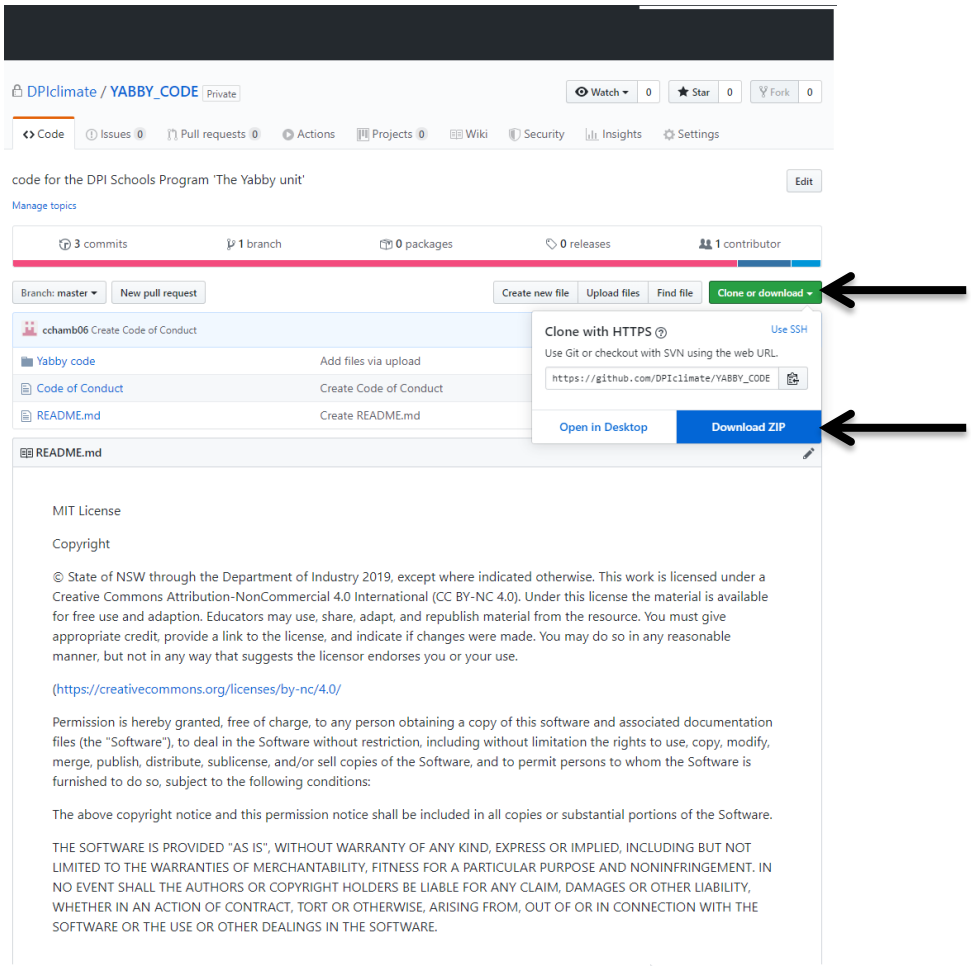
Connect LED green		
28	Connect LED green VIN to Arduino digital pin 8. When writing and manipulating the code, digital pin 8 will control LED green.	
29	Connect part L (330 Ohm resistor) to Bus GND	
Connect LED red		
30	Connect LED Red VIN to Arduino UNO digital pin 10. When writing and manipulating the code, digital pin 10 will control LED red.	
31	Connect part K (220 Ohm resistor) to Bus GND	

Connect LDR (mini photocell) sensor (part E)		
32	Connect part E (LDR) to Arduino UNO analogue pin A1. When writing and manipulating the code, analogue pin A1 will control the LDR sensor.	
33	Connect part E (LDR) to Bus POS	

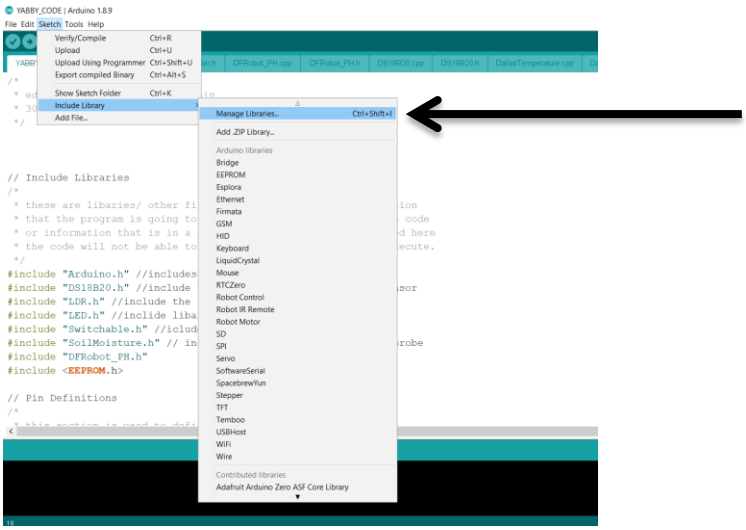
Upload code to your build

The code file for this build is freely available for download from the link at the NSW DPI Schools Program's "The Yabby Unit" page. The following steps are provided as a guide to assist you to upload the code to your build.

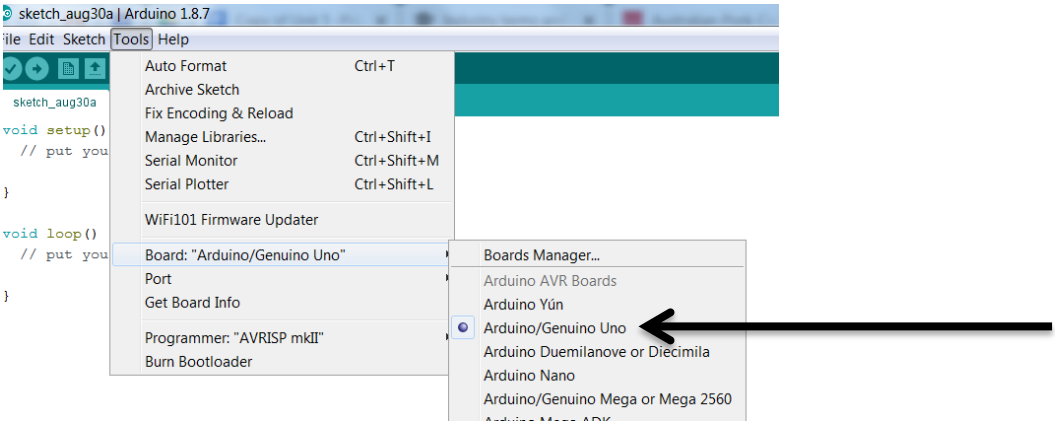
1. Ensure the latest version of Arduino IDE software is downloaded on the computer you plan to use to control the digital system. Arduino IDE software is available for free download at https://www.arduino.cc/en/Main/Software. Follow the steps at the Arduino site to successfully download the IDE software.
2. Go to the NSW DPI Schools Program's "The Yabby Unit" page. Click the button "Download code". This will take you to the DPI Climate teams- Yabby code Github page. Click on the clone or download button, then select Download ZIP.



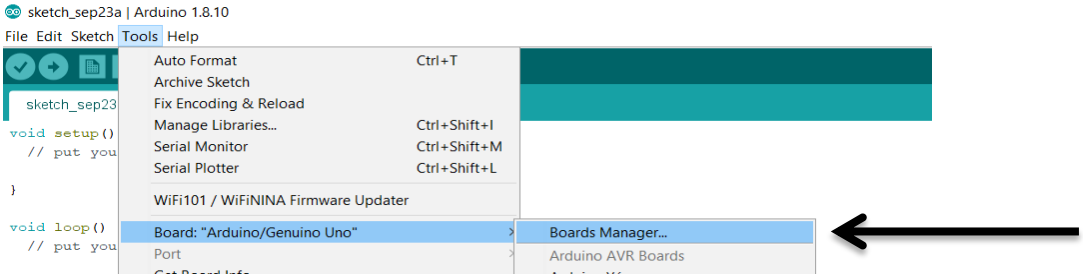
3. Download and unzip the 'Yabby_Code" file using appropriate software.
4. After unzipping the files to your computer; open Arduino IDE and click on Sketch > Include Library > Manage Libraries > Install UNO by arduino.



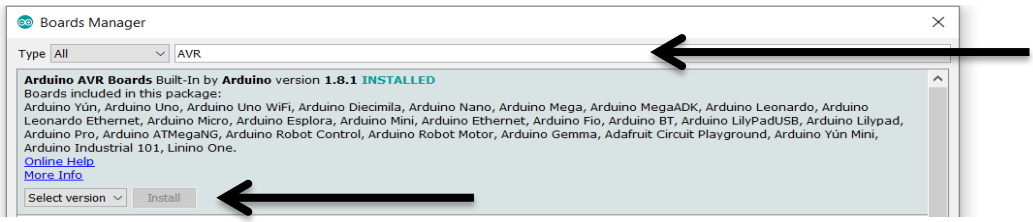
5. In the Arduino IDE program, open Tools > Board > select Arduino/Genuino UNO. (This makes sure you have selected the correct Arduino board to match this build)



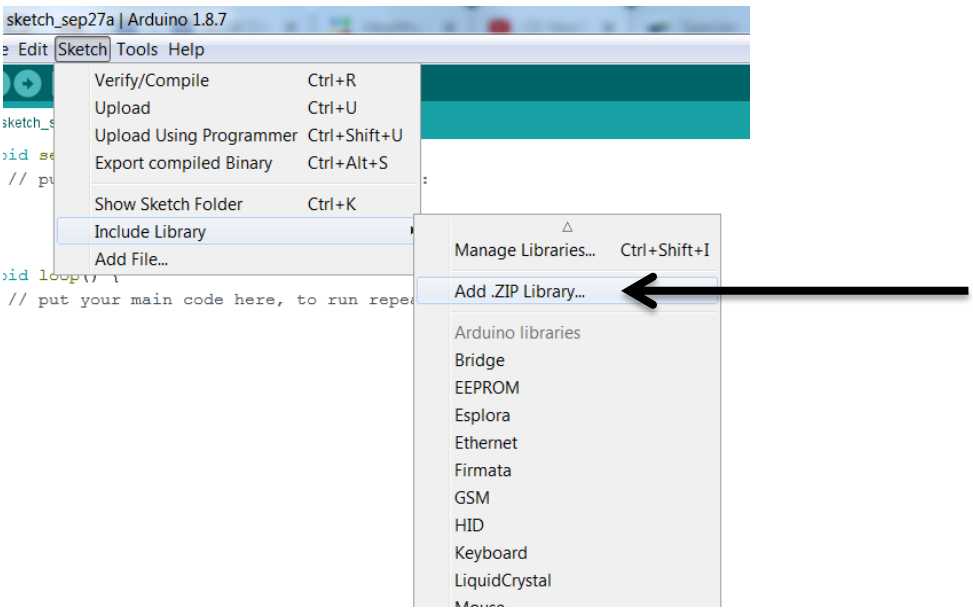
   - If Arduino/Genuino UNO is not available from the step above, open Tools >Board > Boards Manager



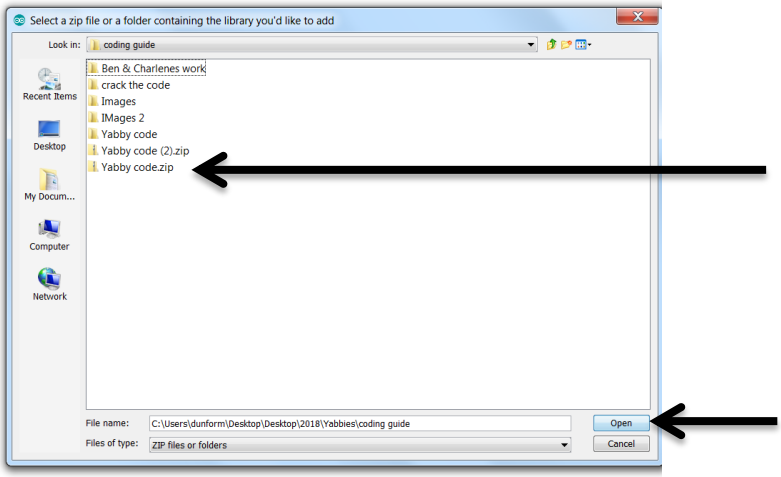
   - In the boards manager search bar type AVR > select Arduino AVR Boards Built-In by Arduino and click Install



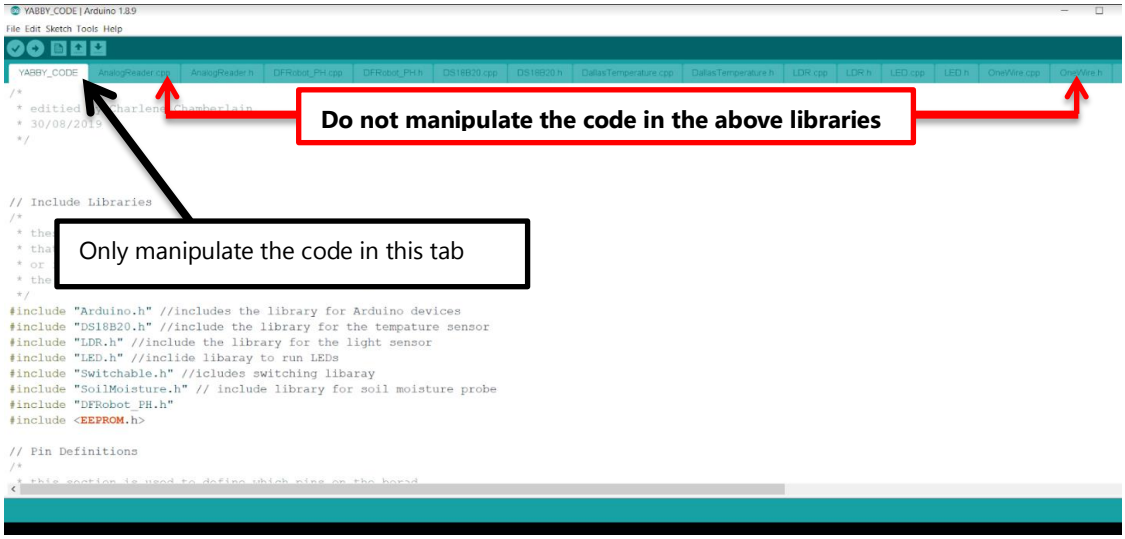
6. Right click on the folder 'YABBY_CODE.ino' and open with Arduino IDE. (This should automatically happen). If not
7. To import the Yabby code, open tab Sketch > Include library > Add. ZIP Library…



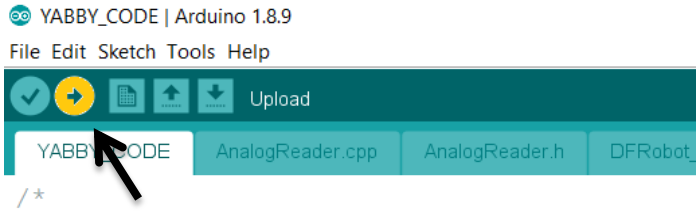
Select you 'YABBY_CODE.ino' file from wherever you saved it on your computer, then select open



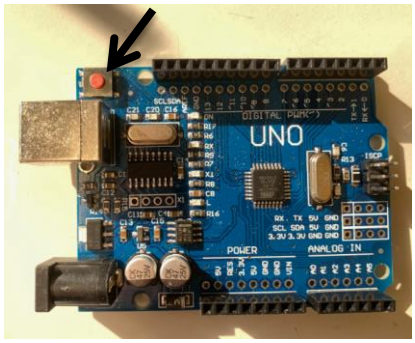
8. Your screen will look like this. This is the sketch menu. The first tab "YABBY_CODE" is the only tab you should manipulate. Do not touch code in the other tabs or the hardware will not work.



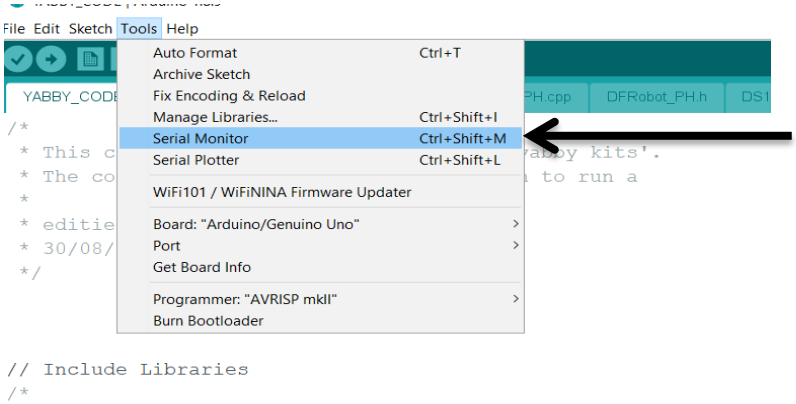
9. Once you have the code in sketch menu, you now need to upload the code to your physical Arduino project build. To do so, click the Upload button (the right-facing arrow icon). This will compile the code and upload it to your circuit build after it passes 'compilation'.



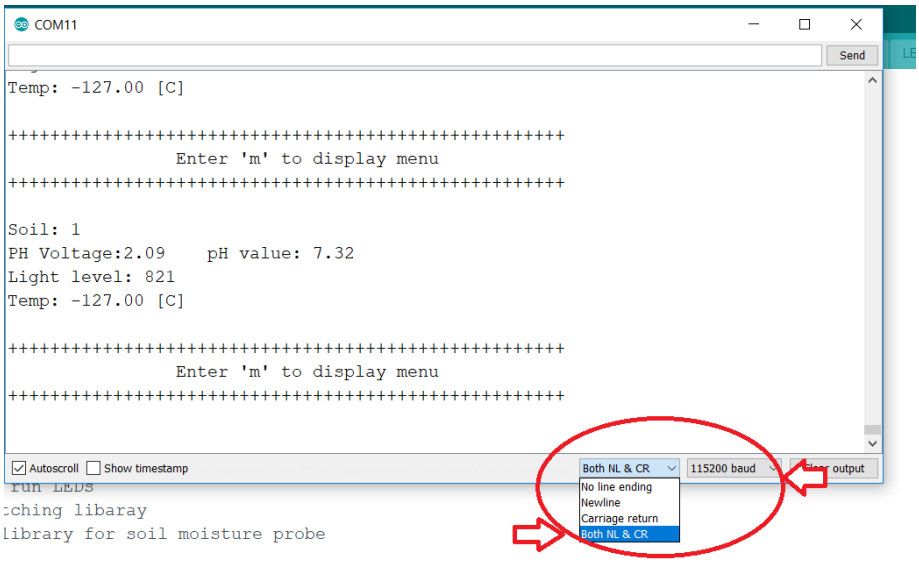
10. If you get any error codes at the base of the screen in red, press the 'reset button' on your Arduino Uno board. And re- do step 8.



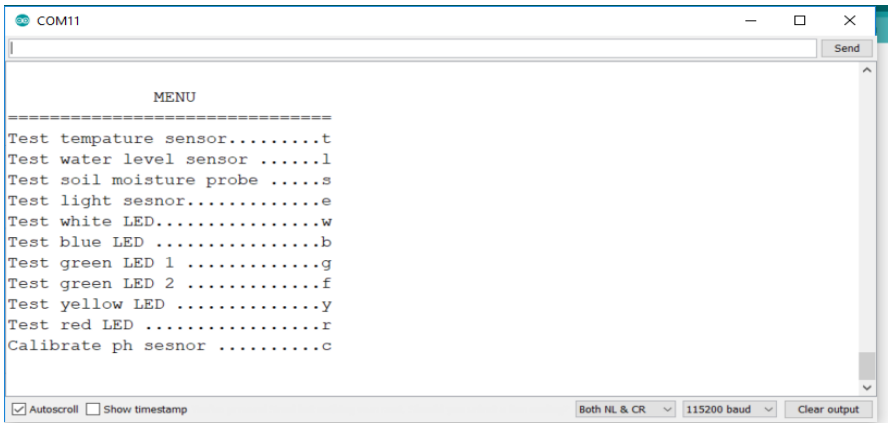
11. Once your upload is complete, the program will start running automatically.
12. All your data readings from your sensors will be displayed in the serial monitor. To access this go to the Tools menu and select Serial monitor.



13. When you have opened the serial monitor for the first time, you must change some settings at the base of the serial monitor screen. Set the 'baud' (bit rate/second) at 115200 and the first box at 'Both NL & CR' for your code to work.



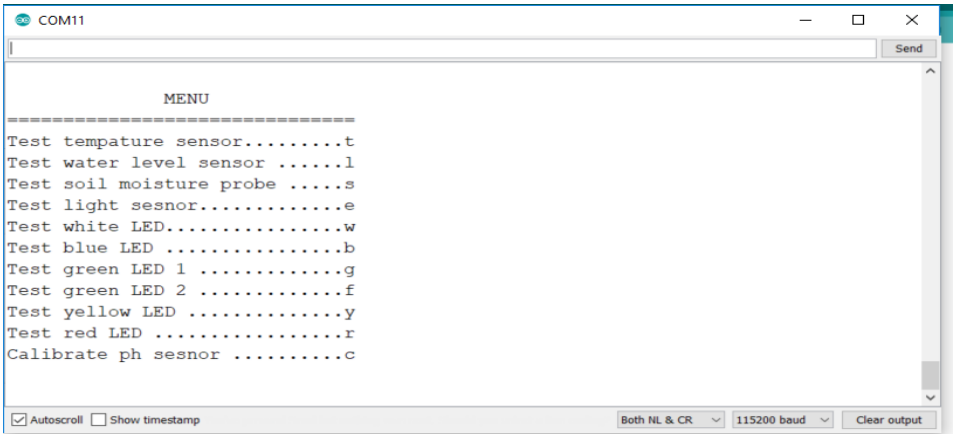
14. Once the program is uploaded and running, in the serial monitor type ‘m’ and hit enter to display the menu. A ‘menu display’ will pop-up that looks like this. The menu is used to test all the physical components.



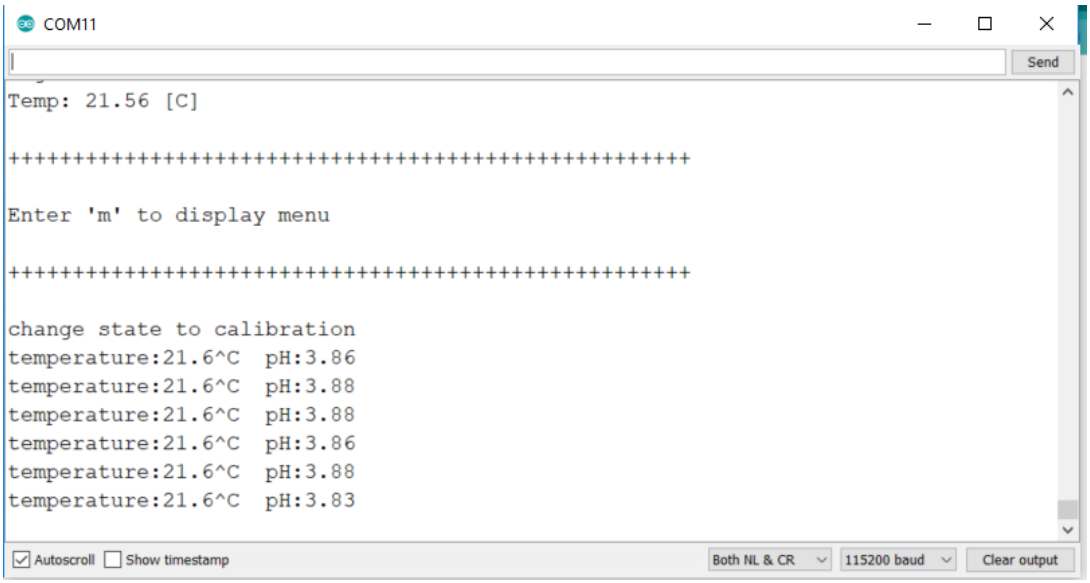
15. Observe readings from the different sensors by using the menu key.

How to calibrate the pH sensor

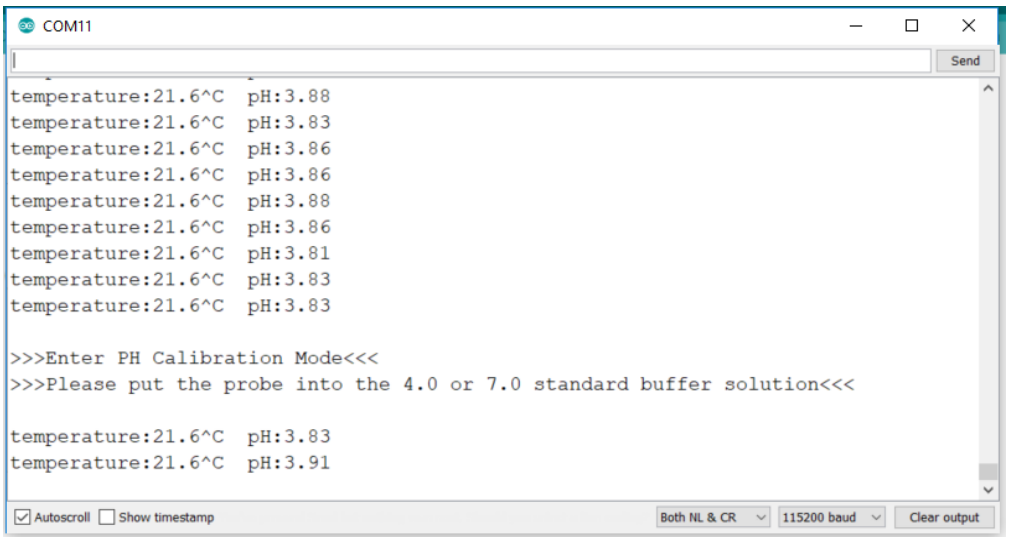
1. Enter the menu by using the 'm' command in the serial monitor. The following menu will 'pop-up'.



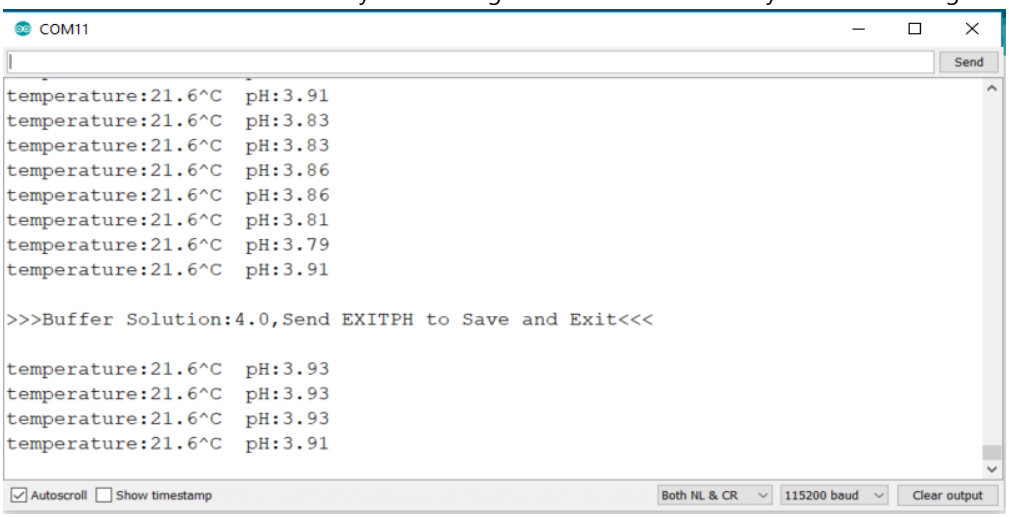
2. Once in the menu enter the calibration state by using the 'c' command. The following menu will 'pop-up'.



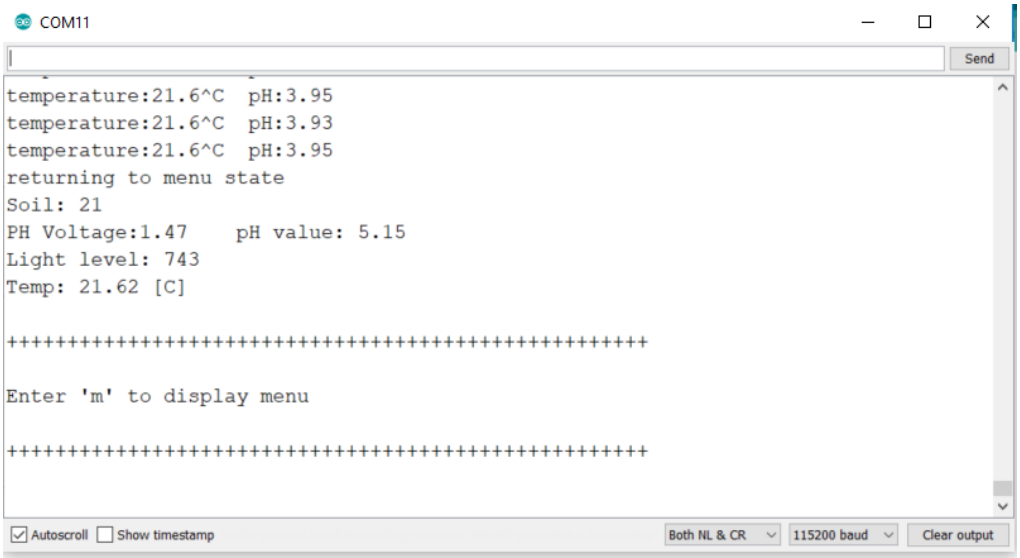
3. Wash the sensor with distilled water, use paper towel to remove any remaining drops of water.
4. Place the sensor in a standard buffer solution (pH 4.0). Stir gently with the sensor until the pH value is consistent.
5. Use the ENTERPH command to enter the calibration mode and follow the instructions.



6. Use the CALPH command to start the pH calibration. This part of the program will automatically identify which of the two buffer solutions you are using. In this case it will identify the 4.0 buffering solution.



7. Use the EXITPH command to save the calibration and exit the calibration mode and state. Only by using the EXITPH command will the calibration be saved.



8. Repeat steps 1-7 using the pH7.0 buffer solution. This will calibrate your pH probe.

Troubleshooting wiring and testing physical components

For more assistance and tips; read "Reading 1- Everything you need to know about Arduino code", in the appendix for more information.

Step	Description
Upload code	The code will allow you to operate each component, and test your writing
Test Water level sensor	1. Make sure the Arduino board is connected to computer via USB cable 2. Open Arduino IDE 3. Click Tools -> Serial Monitor 4. Follow instructions in the Serial Monitor 5. Immerse the water level sensor in water and wait for readings. Remove it and wait for readings. 6. If nothing happens, please check the connections
Test Digital temperature sensor	7. Make sure the Arduino board is connected to computer via USB cable 8. Open Arduino IDE 9. Click Tools -> Serial Monitor 10. Follow instructions in the Serial Monitor 11. Touch the sensor with your fingers to change its temperature reading. Caution if connected incorrectly it may be very hot. 12. If nothing happens, please check the connections
Test soil moisture sensor	1. Make sure the Arduino board is connected to computer via USB cable 2. Open Arduino IDE 3. Click Tools -> Serial Monitor 4. Touch both pads of the sensor with your finger, you should see the values changing 5. If nothing happens check the connections
Test pH sensor	1. Make sure the Arduino board is connected to computer via USB cable 2. Open Arduino IDE 3. Click Tools -> Serial Monitor 4. Touch both pads of the sensor with your finger, you should see the values changing If nothing happens check the connections 5. If using DFROBOT sensor go to the section 'How to calibrate the pH sensor' above for instructions. Otherwise follow this link to the DFROBOT Gravity: Analog pH Sensor calibration instructions
Test LED yellow	1. Make sure the Arduino board is connected to computer via the USB cable 2. Open Arduino IDE 3. Click Tools -> Serial Monitor 4. Follow instructions on the Serial Monitor 5. If nothing happens check the connections.
Test LED blue	1. Make sure the Arduino board is connected to computer via the USB cable 2. Open Arduino IDE 3. Click Tools -> Serial Monitor 4. Follow instructions on the Serial Monitor 5. If nothing happens check the connections.
Test LED green	1. Make sure the Arduino board is connected to computer via the USB cable

	2. Open Arduino IDE 3. Click Tools -> Serial Monitor 4. Follow instructions on the Serial Monitor 5. If nothing happens check the connections.
Test LED red	1. Make sure the Arduino board is connected to computer via the USB cable 2. Open Arduino IDE 3. Click Tools -> Serial Monitor 4. Follow instructions on the Serial Monitor 5. If nothing happens check the connections.
Test LDR (mini photocell) sensor	1. Make sure the Arduino board is connected to the computer via USB cable 2. Open Arduino IDE 3. Click Tool -> Serial Monitor 4. Follow instructions on the Serial Monitor 5. If nothing happens, please check the connections

References and further reading

4 simple steps for debugging your Arduino project

Zait, A, (2018) *'4 simple steps for debugging your Arduino project'*, Circuito.io blog, Roboplan Technologies Ltd, https://www.circuito.io/blog/arduino-debugging/, viewed 19 December 2018

An introduction to Arduino pinout

Zait, A, (2018) *'An introduction to Arduino pinout'*, Circuito.io blog, Roboplan Technologies Ltd, https://www.circuito.io/blog/arduino-uno-pinout/, viewed 19 December 2018

Cicuito.io

Roboplan Technologies Ltd, 2018, *'Circuito.io'*, Roboplan Technologies Ltd, www.circuito.io/, viewed 7 December 2018

Crack the Code

Department of Education, 2018, '*Crack the Code*', Sate of NSW, Department of Education, NSW Government, Education, Public Schools, https://education.nsw.gov.au/teaching-and-learning/curriculum/key-learning-areas/tas/s4-5/resources/crack-the-code, viewed July 24 2018

Everything you need to know about Arduino code

Circuito Team, (2018) 'Everything you need to know about Arduino code', Circuito.io blog, Roboplan Technologies Ltd, https://www.circuito.io/blog/arduino-code/,, viewed 19 December 2018

How Arduino sensors actually work

Zait, A, (2018) *'How Arduino sensors actually work'*, Circuito.io blog, Roboplan Technologies Ltd, https://www.circuito.io/blog/arduino-sensors-explained/, viewed 19 December 2018

Tinkercad Circuits

Tinkercad, 2018, '*Tinkercad circuits*', Tinkercad, https://www.tinkercad.com/circuits, viewed 22 October 2018

Appendix

Reading 1 "Everything you need to know about Arduino code"

Circuito Team, (2018) '*Everything you need to know about Arduino code*', Circuito.io blog, Roboplan Technologies Ltd, https://www.circuito.io/blog/arduino-code/,, viewed 19 December 2018

EVERYTHING YOU NEED TO KNOW ABOUT ARDUINO CODE

BLOG POST CIRCUITO TEAM MARCH 11, 2018

Since the launch of the Arduino open-source platform, the brand has established themselves at the center of an expansive open-source community. The Arduino ecosystem is comprised of a diverse combination of hardware and software. The versatility of Arduino and its simple interface makes it a leading choice for a wide range of users around the world from hobbyists, designers, and artists to product prototypes.

The Arduino board is connected to a computer via USB, where it connects with the Arduino development environment (IDE). The user writes the Arduino code in the IDE, then uploads it to the microcontroller which executes the code, interacting with inputs and outputs such as sensors, motors, and lights.

Both beginners and experts have access to a wealth of free resources and materials to support them. Users can look up information on how to set up their board or even how to code on Arduino. The open source behind Arduino has made it particularly friendly to new and experienced users. There are thousands of Arduino code examples available online. In this post, we'll take you through some basic principles of coding for Arduino.

Arduino Coding Environment and basic tools

What language is Arduino?

Arduino code is written in C++ with an addition of special methods and functions, which we'll mention later on. C++ is a human-readable programming language. When you create a 'sketch' (the name given to Arduino code files), it is processed and compiled to machine language.

Arduino IDE

The Arduino Integrated Development Environment (IDE) is the main text editing program used for Arduino programming. It is where you'll be typing up your code before uploading it to the board you want to program. Arduino code is referred to as **sketches**.

Note: It's important to use the latest version of Arduino IDE. From time to time, check for updates here.

Arduino code example

As you can see, the IDE has a minimalist design. There are only 5 headings on the menu bar, as well as a series of buttons underneath which allow you to verify and upload your sketches. Essentially, the IDE translates and compiles your sketches into code that Arduino can understand. Once your Arduino code is compiled it's then uploaded to the board's memory.

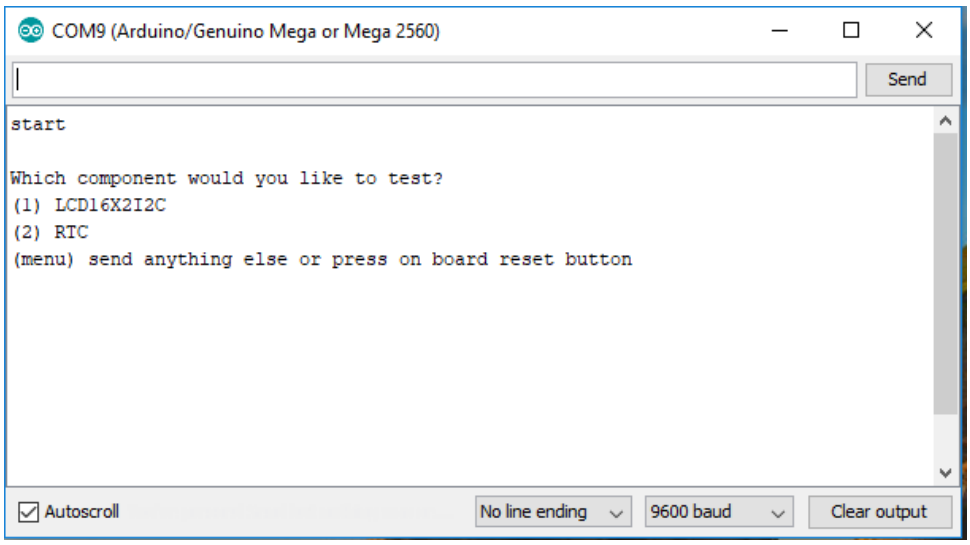
All the user has to do to start compiling their sketch is press a button (a guide to this can be found below).

If there are any errors in the Arduino code a warning message will flag up prompting the user to make changes. Most new users often experience difficulty with compiling because of Arduino's stringent syntax requirements. If you make any mistakes in your punctuation when using Arduino, the code won't compile and you'll be met with an error message.

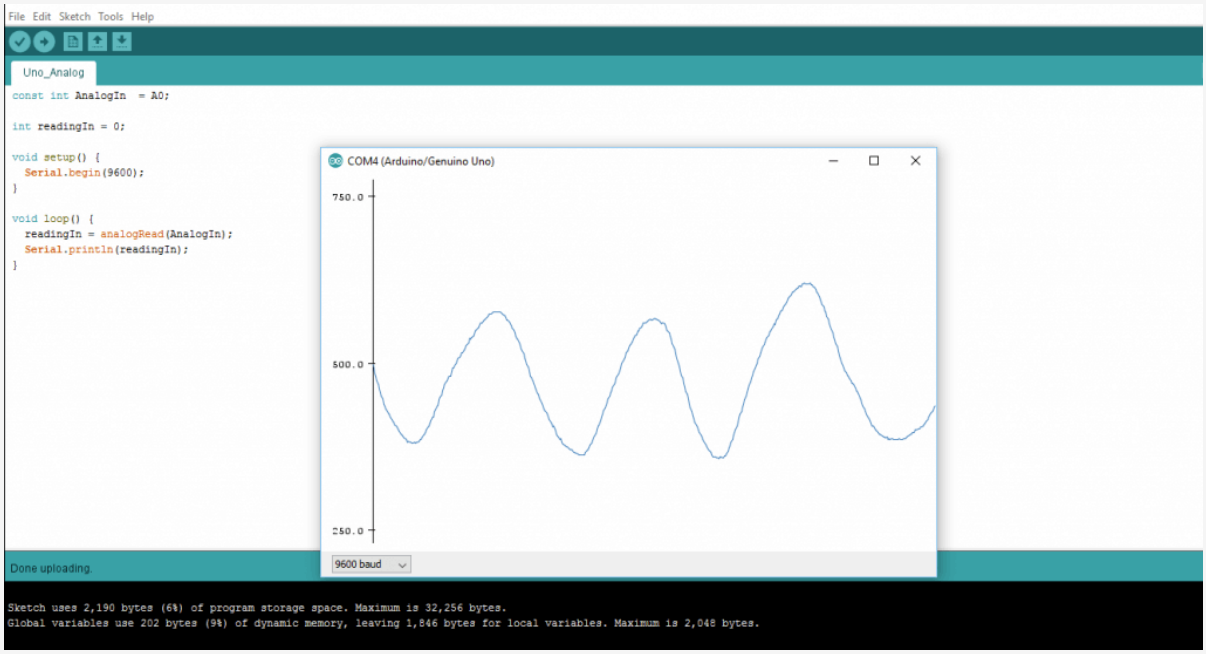
Serial Monitor and Serial Plotter

Arduino **serial monitor** can be opened by clicking on the magnifying glass icon on the upper right side of the IDE or under tools. The serial monitor is used mainly for interacting with the Arduino board using the computer, and is a great tool for real-time monitoring and debugging. In order to use the monitor, you'll need to use the Serial class.

The code you download from circuito.io has a test section that helps you test each components using the serial monitor, as you can see in the screenshot below:



Arduino **serial plotter** is another component of the Arduino IDE, which allows you to generate a real-time graph of your serial data. The serial plotter makes it much easier for you to analyze your data through a visual display. You're able to create graphs, negative value graphs, and conduct waveform analysis.



Debugging Arduino Code and Hardware

Unlike other software programming platforms, Arduino doesn't have an onboard debugger. Users can either use third-party software, or they can utilize the serial monitor to print Arduino's active processes for monitoring and debugging.

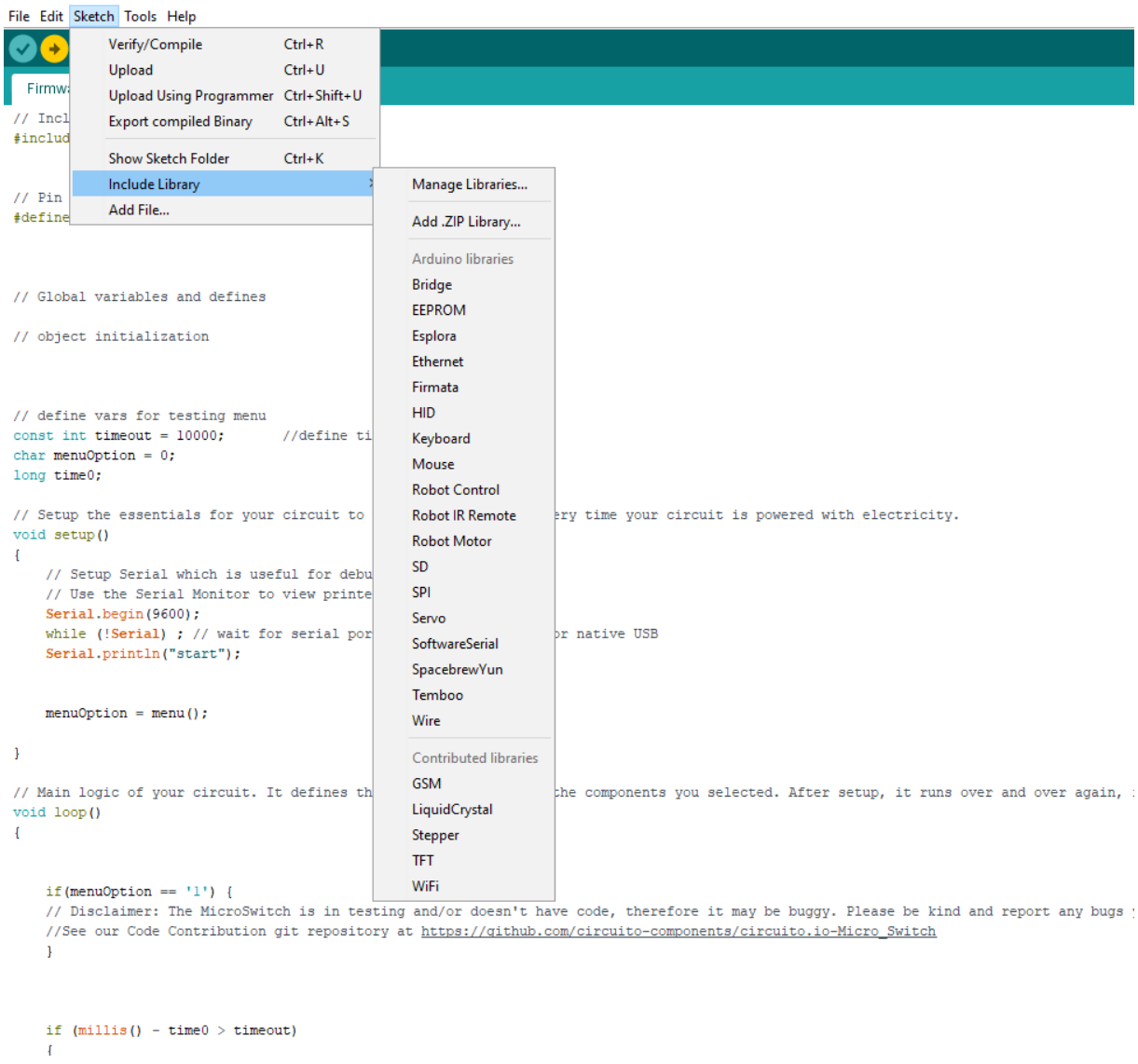
By using the Serial class, you can print to the serial monitor, debugging comments and values of variables. On most Arduino models, this will be using serial pins 0 and 1 which are connected to the USB port.

Code Structure

Libraries

In Arduino, much like other leading programming platforms, there are built-in libraries that provide basic functionality. In addition, it's possible to import other libraries and expand the Arduino board capabilities and features. These libraries are roughly divided into libraries that interact with a specific component or those that implement new functions.

To import a new library, you need to go to Sketch > Import Library



In addition, at the top of your .ino file, you need to use '#include' to include external libraries. You can also create custom libraries to use in isolated sketches.

Pin Definitions

To use the Arduino pins, you need to define which pin is being used and its functionality. A convenient way to define the used pins is by using:

'#define pinName pinNumber'.

The functionality is either input or output and is defined by using the pinMode () method in the setup section.

Declarations

Variables

Whenever you're using Arduino, you need to declare global variables and instances to be used later on. In a nutshell, a variable allows you to name and store a value to be used in the future. For example, you would store data acquired from a sensor in order to use it later. To declare a variable you simply define its type, name and initial value.

It's worth mentioning that declaring global variables isn't an absolute necessity. However, it's advisable that you declare your variables to make it easy to utilize your values further down the line.

Instances

In software programming, a **class** is a collection of functions and variables that are kept together in one place. Each class has a special function known as a **constructor**, which is used to create an **instance** of the class. In order to use the functions of the class, we need to declare an instance for it.

Setup()

Every Arduino sketch must have a setup function. This function defines the initial state of the Arduino upon boot and runs only once.

Here we'll define the following:

1. Pin functionality using the pinMode function
2. Initial state of pins
3. Initialize classes
4. Initialize variables
5. Code logic

Loop()

The loop function is also a must for every Arduino sketch and executes once setup() is complete. It is the main function and as its name hints, it runs in a loop over and over again. The loop describes the main logic of your circuit.

For example:

Note: The use of the term ‘void’ means that the function doesn’t return any values.

How to program Arduino

The basic Arduino code logic is an “if-then” structure and can be divided into 4 blocks:

Setup - will usually be written in the setup section of the Arduino code, and performs things that need to be done only once, such as sensor calibration.

Input - at the beginning of the loop, read the inputs. These values will be used as conditions (“if”) such as the ambient light reading from an LDR using analogRead().

Manipulate Data - this section is used to transform the data into a more convenient form or perform calculations. For instance, the AnalogRead() gives a reading of 0-1023 which can be mapped to a range of 0-255 to be used for PWM.(see analogWrite())

Output - this section defines the final outcome of the logic (“then”) according to the data calculated in the previous step. Looking at our example of the LDR and PWM, turn on an LED only when the ambient light level goes below a certain threshold.

Arduino Code libraries

Library Structure

A library is a folder comprised of files with C++ (.cpp) code files and C++ (.h) header files.

The .h file describes the structure of the library and declares all its variables and functions.

The .cpp file holds the function implementation.

Importing Libraries

The first thing you need to do is find the library you want to use out of the many libraries available online. After downloading it to your computer, you just need to open Arduino IDE and click on Sketch > Include Library > Manage Libraries. You can then select the library that you want to import into the IDE. Once the process is complete the library will be available in the sketch menu.

In the code provided by circuito.io instead of adding external libraries like mentioned before, we provide them with the firmware folder. In this case, the IDE knows how to find them when using #include.

From Software to Hardware

There is a lot to be said of Arduino's software capabilities, but it's important to remember that the platform is comprised of both software and hardware. The two work in tandem to run a complex operating system.

Code → Compile → Upload → Run

At the core of Arduino, is the ability to compile and run the code.

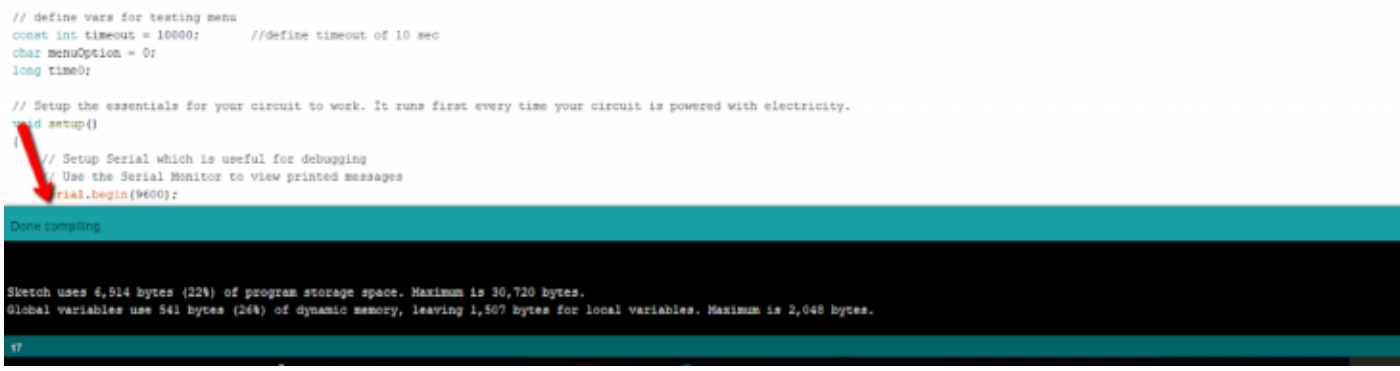
After writing the code in the IDE you need to upload it to the Arduino. Clicking the Upload button (the right-facing arrow icon), will compile the code and upload it if it passed compilation. Once your upload is complete, the program will start running automatically.

You can also do this step by step:

1. First, compile the code. To do this simply click the check icon (or click on sketch > Verify / Compile in the menu bar.

As you can see, the check icon is located in the top left underneath the "File" tag in the menu section.

Once you've done this, Arduino will start to compile. Once it's finished, you'll receive a completion message that looks like this:



As you can see, the green line at the bottom of the page tells you that you're "done compiling". If your code fails to run, you'll be notified in the same section, and the problematic code will be highlighted for editing.

Once you've compiled your sketch, it's time to upload it.

1. Choose the serial port your Arduino is currently connected to. To do this click on Tools > Serial port in the menu to designate your chosen serial port (as shown earlier above). You can then upload the compiled sketch.
2. To upload the sketch, click on the upload icon next to the tick. Alternatively you can go the the menu and click File> upload. Your Arduino LEDS will flicker once the data is being transferred.

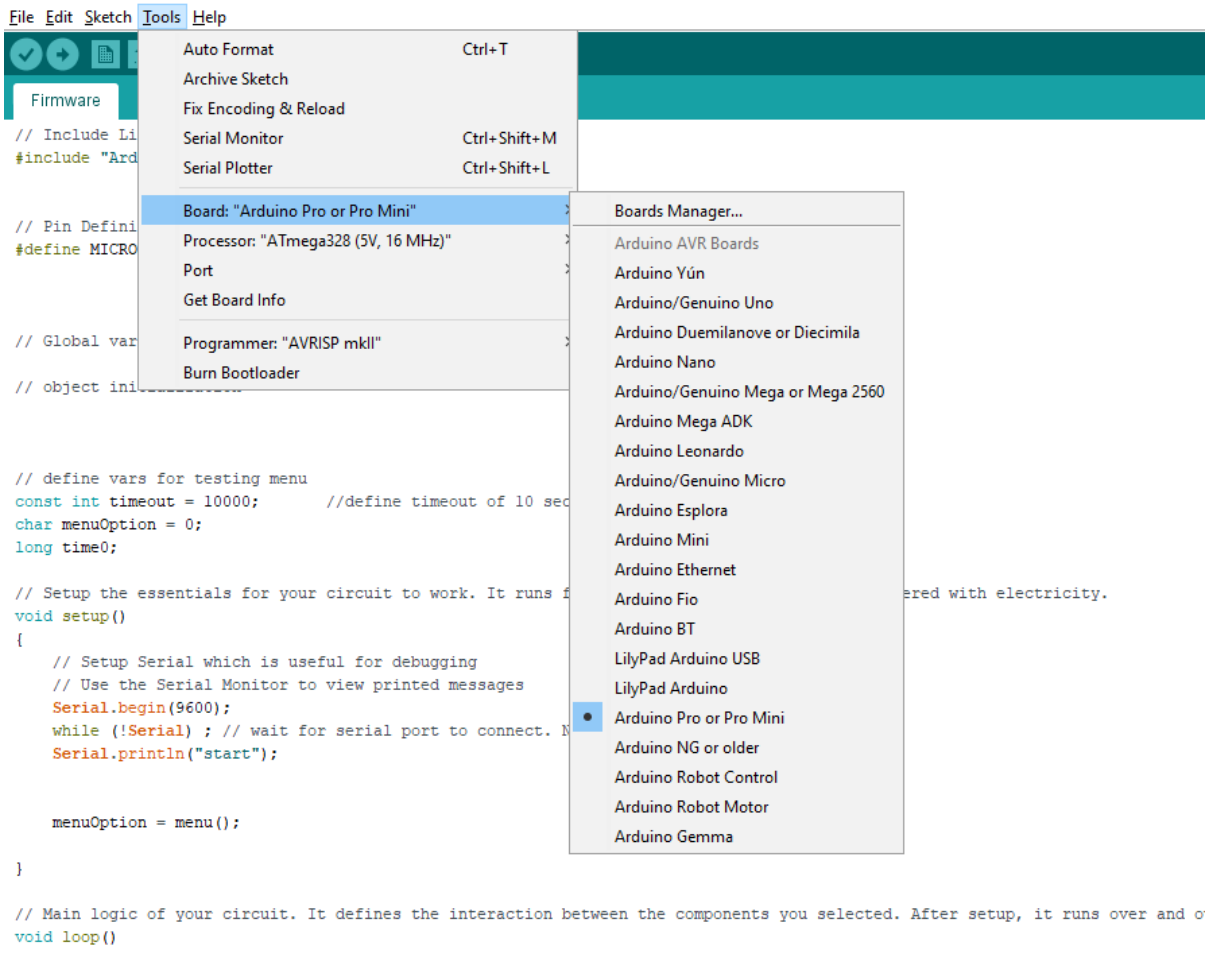
Once complete, you'll be greeted with a completion message that tells you Arduino has finished uploading.

Setting Up Your IDE

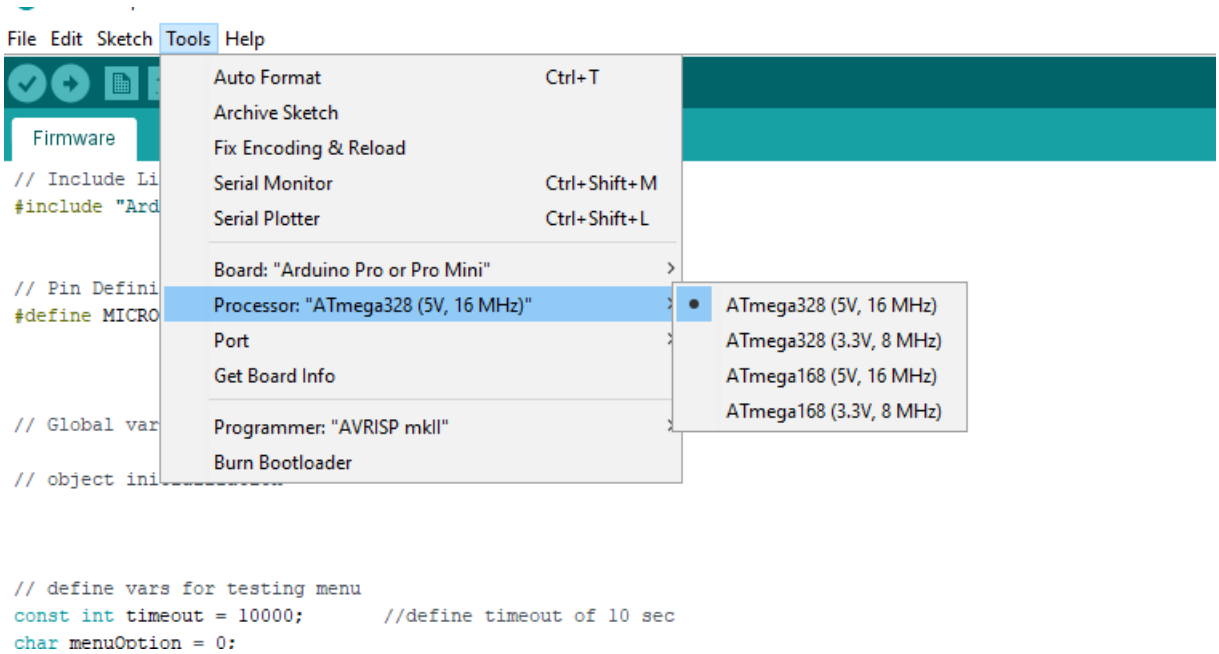
In order to connect an Arduino board to your computer you need a USB cable. When using the Arduino UNO, the USB transfers the data in the program directly to your board. The USB cable is used to power your arduino. You can also run your Arduino through an external power source.

Before you can upload the code, there are some settings that you need to configure.

Choose your board - You need to designate which Arduino board you're going to be using. Do this by click Tools > Board > Your Board.



Choose your processor- there are certain boards (for example Arduino pro-mini) for which you need to specify which processor model you have. Under tools > processor > select the model you have.



Choose your port - to select the port to which your board is connected, go to tools > Port > COMX Arduino (This is Arduino's serial port).

How to Install Non-Native Boards (e.g. NodeMCU)

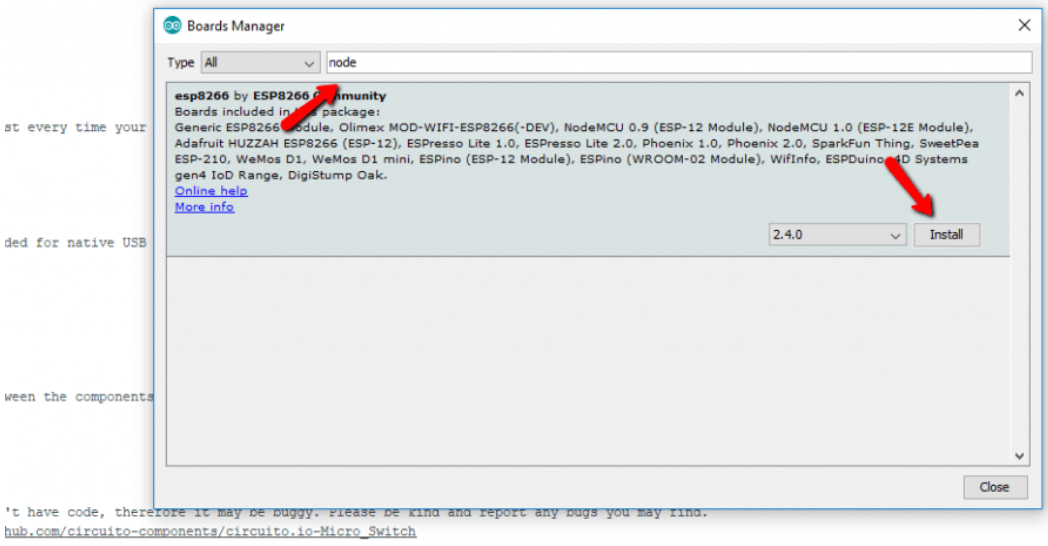
Some board models are not pre-installed in the Arduino IDE, therefore you'll need to install them before you can upload code.

To install a non-native board such as NodeMCU, you need to:

1. Click on tools > Boards > Boards Manager
2. Search for the board you want to add in the search bar and click "install".

Some boards cannot be found through the Board Manager. In this case, you'll need to add them manually. In order to do this:

1. Click on Files > Preferences
2. In the Additional Boards Manager field, paste the URL of the installation package of your board. For instance, for nodeMCU, add the following URL: http://arduino.esp8266.com/stable/package_esp8266com_index.json
3. Click OK
4. Go to tools > Boards > Boards Manager
5. Search for the board you want to add in the search bar and click "install".



Once you've completed this step, you will see the installed boards in the boards' list under tools.

Note: the process may differ slightly for different boards.

Arduino: An Extremely Versatile Platform

Arduino is far more than just a simple microcontroller. With an expansive IDE and a vast array of hardware configurations, Arduino is truly a diverse platform. The variety of its libraries and its intuitive design makes it a favorite for new users and experienced makers alike. There are thousands of community resources to help you get started with both hardware and software.

As you advance your skills, you may face problems that require debugging, which is a weak spot of the Arduino IDE. Luckily, there are several tools and methods to debug Arduino hardware and software. In the next article, we're going to look at how to debug Arduino (and how to test Arduino code) as well as how to use simulators and emulators.

Reading 2 "How Arduino sensors actually work"

Zait, A, (2018) *'How Arduino sensors actually work'*, Circuito.io blog, Roboplan Technologies Ltd, https://www.circuito.io/blog/arduino-sensors-explained/, viewed 19 December 2018

HOW ARDUINO SENSORS ACTUALLY WORK

BLOG POST ANAT ZAIT TUE DEC 04 2018

If you're going to use your Arduino to its fullest potential, then you'll need to pair it with one or more extra components. After all, an Arduino that's incapable of interacting with the world around it isn't much fun!

Components in an electronic circuit can be divided into three categories:

+ Input devices, like buttons, switches, and sensors.

+ Processors, such as standalone microcontrollers and single board computers like the Arduino, ESP and Raspberry Pi.

+ Outputs (or actuators), which convert those voltages into sound waves or lights that we can sense.

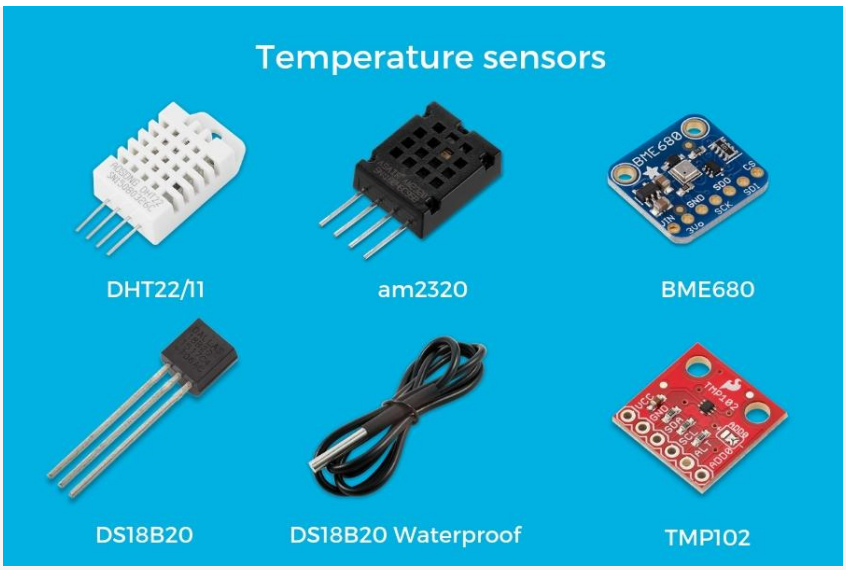
In this post, we'll focus on the first part of this chain. All input devices, can be classed as sensors. These are components whose output changes according to a property of the surrounding environment. This can be anything from a change in the temperature, to gravity, to electromagnetic charge. Sensors are incredibly useful devices. Without them, electrical circuitry just wouldn't make much sense!

If you have an Arduino, then you'll find it easy to incorporate a sensor or two into your projects. But with so many to choose from, deciding which device to use may not be so simple.

Sensors come in all sorts of shapes and sizes. Here, we'll examine five different types in detail and try to determine how (and why) we might implement each of them into our designs. We'll also talk about where they can be used, what their advantages and drawbacks are, and hopefully, provide you with a little bit of inspiration for your project. Let's begin!

Temperature sensors

A temperature sensor, as you can probably guess, detects changes in temperature. They're a key component in any device that relies on this information, like an automated gardening system which opens or closes a window in a greenhouse depending on the temperature inside.

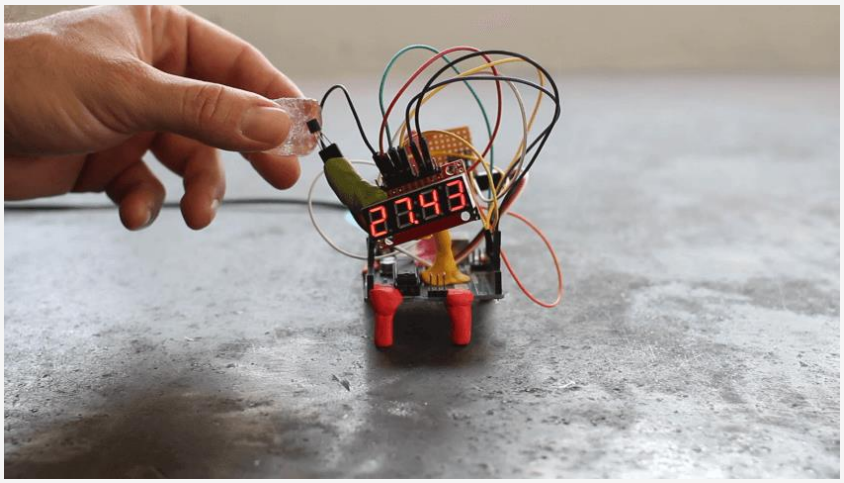


Thermostat

Among the most common sorts of temperature sensor is the thermostat. Most thermostats contain two metallic strips, placed one on top of the other. One changes shape in response to changes in temperature, causing both to bend and touch a nearby contact. Thermostats are inexpensive to produce, but they aren't as accurate as other sorts of temperature-sensitive sensors, like thermistors.

[images: examples of thermostats]

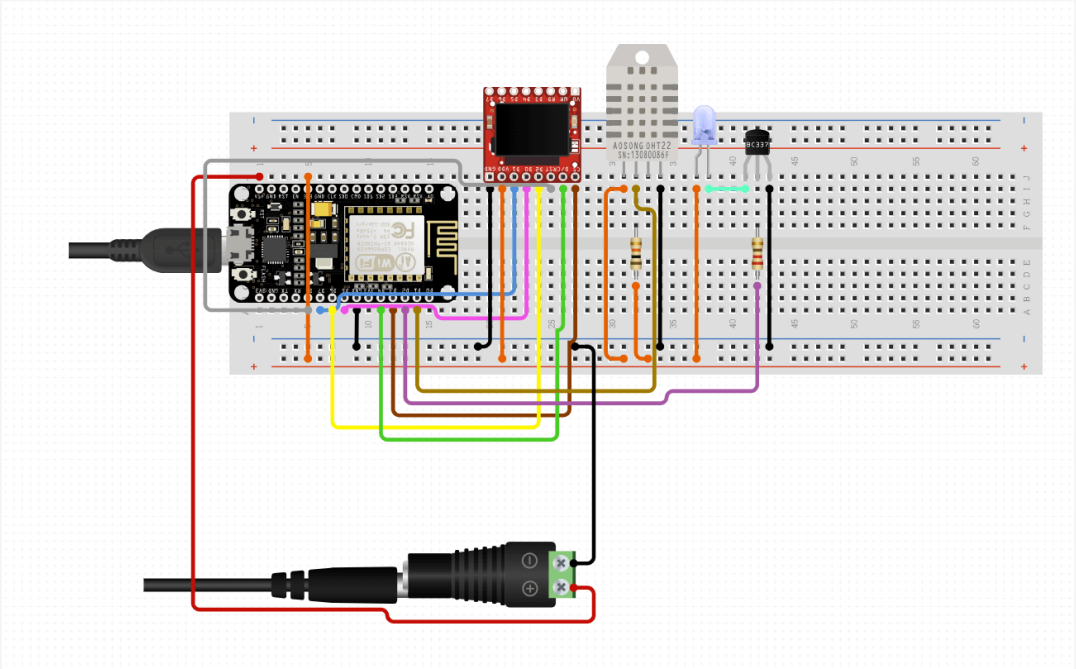
The numeric readout provided by our DIY digital thermometer is a simple application of the temperature sensor. If you can keep it moisture-tight, you'll be able to leave it in your refrigerator, and see at a glance how cold it is inside. This device puts out a digital reading along a single wire, and doesn't even need to be powered along a separate one, making it a convenient choice.



Thermistors

All conductive material become less conductive when exposed to heat. A thermistor exploits this phenomenon in a controlled way, and allows a circuit to figure out the temperature by recording the resistance. Thermistors come in two sorts: negative temperature coefficient (NTC) and positive temperature coefficient (PTC). The former offers less resistance as the temperature rises; the latter offers more. By choosing the right one, you'll be able to keep your circuit simple.

Our NodeMCU-powered smart-thermostat incorporates the DHT22, which uses a thermistor to measure temperature, and another embedded sensor to measure humidity. The two sensors are connected internally to a microcontroller which outputs a digital signal along a single wire. This extra stuff slows the circuit down a little, with samples being taken every two seconds – for our purposes, however, this wasn't a problem.



The DHT22 also features prominently in our Arduino-based weather station project, which can be used to submit your local temperature to a wider network of public and private weather stations, helping everyone in the process.

Thermocouples

A thermocouple is made from a couple (get it?) of junctions of different metals, joined together. The resistance of one will alter differently to the other, so as the heat rises, the difference in voltage between the ends of the two will change.

A thermocouple is capable of measuring far greater ranges in temperature than a thermistor, which makes them suitable for high-stress environments, like the interior of a gas turbine or an engine. This advantage, however, comes at the cost of accuracy: thermocouples are nowhere near as accurate as thermistors.

RTD

A Resistance Temperature Detector, or RTD, works pretty much like a thermistor, except that it's made from a coil of highly conductive metal, whose conductivity changes in response to the local temperature. They represent a compromise between the thermocouple and the thermistor, in that they can measure more accurately than the thermocouple, over a greater range than the thermistor. Where the RTD falls down is response time; while the other components we've mentioned will react to temperature changes within a few seconds, the RTD can sometimes take up to a minute. As such, it's better at observing longer trends than reacting to short-term thermal shocks.

Applications for temperature sensors

So where are these various technologies applied? A more common example of a temperature sensor is one you probably already have in your home – a thermostat. This device uses a temperature sensor(or several), to track the temperature in a house, and uses the information collected to turn the heating up or down. Without it, the boiler wouldn't know whether to boil more water, and it would be impossible to keep the temperature consistent.

The same principles apply to your oven – built-in temperature sensors inside the compartment constantly check to see whether the temperature is above or below the desired level. If things are too warm or too cold, it'll adjust the power going to the heating elements. Similar arrangements exist in other appliances where temperature needs to be controlled – including fridges, air-conditioning units, the nozzle on your 3D printer, the tip of your soldering iron and the CPU on your computer.

How to Choose a Temperature Sensor

Accuracy

Of the components we've mentioned, a thermistor is by far the most accurate.

Range

In a thermocouple, the actual sensor is shielded, leaving only the metal arms to which it's attached exposed to higher temperatures. As such, it's a great choice for high-temperature situations.

Interface (analog/digital)

Pretty much all temperature sensors put out an analog current; you'll need to use an ADC to turn their outputs into a useable digital value.

Response Time

If you're measuring very rapid changes in temperature, then you need an accordingly fast response time. In this respect, thermistors and thermocouples come out on top; they're able to respond within just a few hundred milliseconds, while an RTD might take ten times longer or more.

Distance sensors

As with temperature, there are several electronic means of measuring distance. For the most part, they work in a similar way: by putting out an impulse and examining the return signal for changes, much like echolocation.

These sensors use the Time of Flight (ToF) principle, and calculate distance using the following equation:

$$d = \frac{c \times t}{2}$$

d - calculated distance [*meter*] t - measured time [*seconds*] c - speed [$\frac{meter}{second}$]

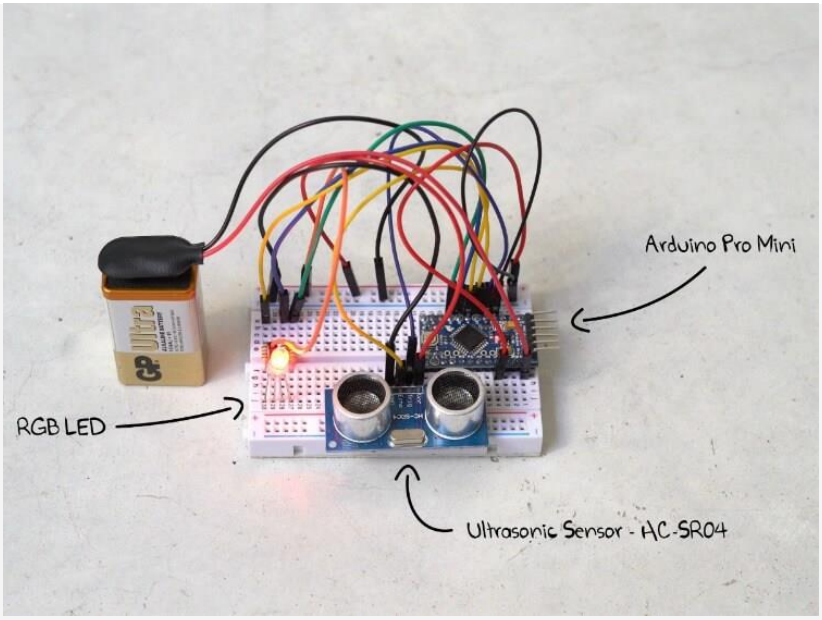
LIDAR

LIDAR (or light-detection and ranging) uses a laser to send a stream of pulses toward a distant target. The delay between the impulse and the reflection arriving back at the sensor,along with the speed of light (which is 299,792,458 metres per second) can be used to work out the distance using the equation above. By doing this thousands of times a second in multiple directions, it's possible to generate a three-dimensional image of the area. In this way, very detailed and accurate maps can be created without the need for an on-site survey team. Lidar can generate images of crime scenes for later study by forensic scientists, detect the speed of passing vehicles, and even generate terrain data for use in video games.

Ultrasonic sensors

Ultrasonic sensors do the same thing using high-frequency sounds rather than light. They pair an ultrasonic emitter with a miniature microphone designed to detect the reflections. They don't draw much current, but the information they provide is of a low resolution and refresh rate. And, naturally, they're able to operate in the dark. In this case, distance is calculated using the ToF equation with the speed of sound travelling through air. Given that this varies according to things like humidity, pressure and temperature, ultrasonic sensors aren't quite as accurate as LIDAR - particularly over long distances.

We've used the HC-SR04 ultrasonic sensor in our Arduino-based air-gate project. It'll detect whenever a drone passes close by, and change color accordingly. Perfect for practicing your piloting skills!



IR LED sensors

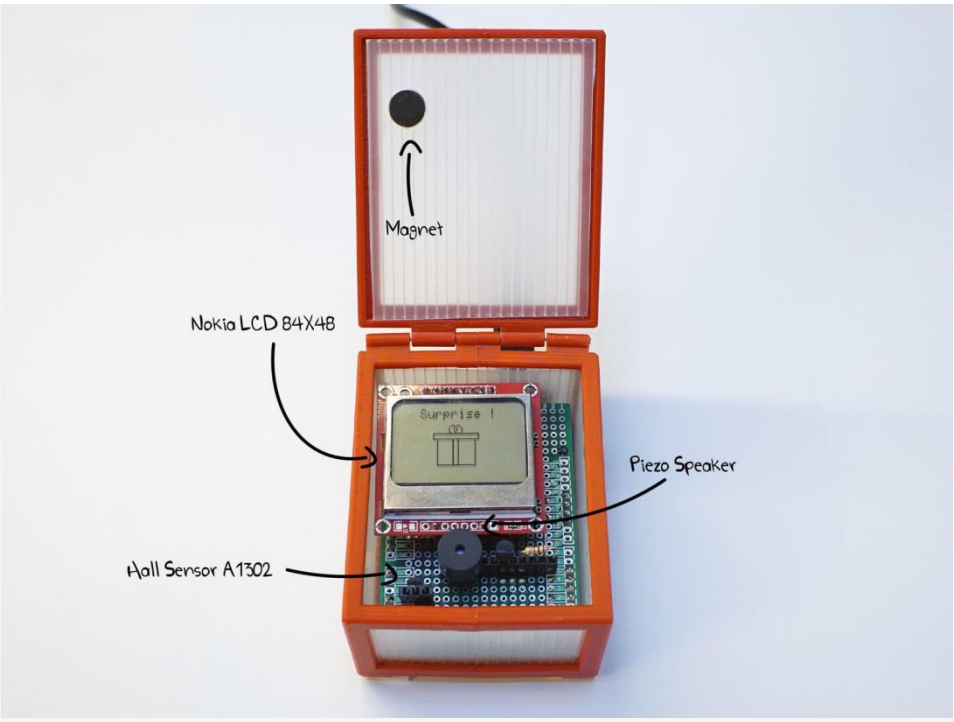
Infrared LED sensors work using light-emitting diodes. Their maximum range is pretty limited. As such, they're ideal for measuring very short distances. A great example would be in 3d printing, where an LED sensor is used to measure (and compensate for) minute changes in the distance between the print-surface and the print-head.

Hall effect proximity sensors

Hall effect sensors put out a voltage depending on the surrounding magnetic field, thereby exploiting a principle discovered by Edwin Hall in the late 19th century. Hall effect sensors are widespread in certain sorts of switches, which have a tiny magnet built into the underside of a key.

One of the advantages of hall-effect sensors is that they don't require physical contact between the parts of the switch. This drastically reduces mechanical wear-and-tear. For this reason, you can find hall effect sensors in many applications, from limit switches in 3d printers, to bicycle speedometers.

If you'd like to get to grips with Hall-Effect sensors and what they can do, then be sure to check out our hall-effect-powered gift box.



How to choose a distance sensor

Application

Our first consideration should be where the component will be used. There's a considerable potential for overkill, here; you don't need LIDAR to know when someone's passed through a door.

Accuracy

LIDARs and IR LED sensors offer excellent precision at long and very close range, respectively.

Range/Distance

LED and hall effect sensors are good for short distances. If you want to measure across an entire room, an Ultrasonic or LIDAR sensor is probably appropriate.

Footprint

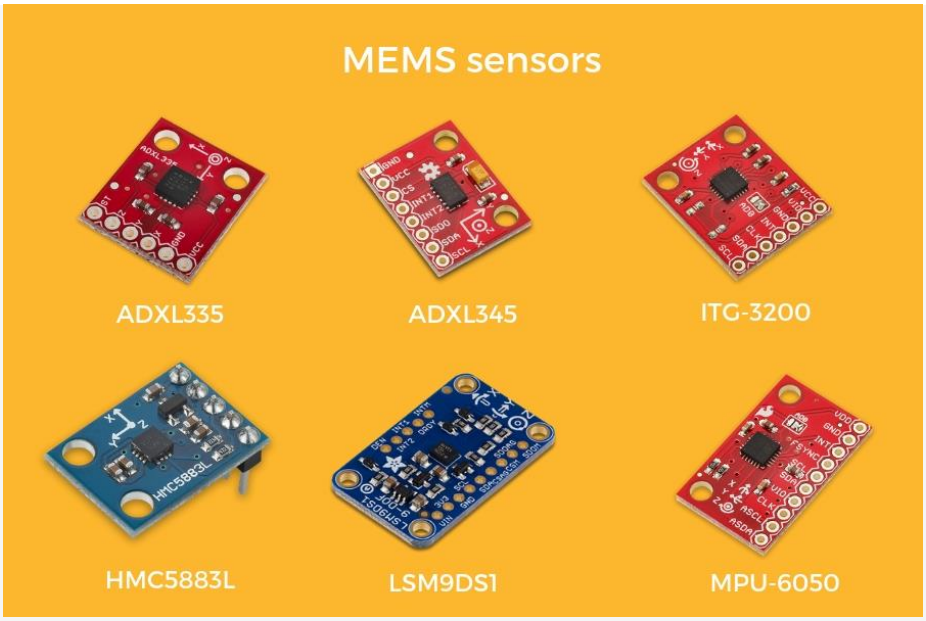
The average LIDAR is bigger than an Arduino, and thus makes a poor match for very compact projects, where LED and Hall Effect sensors thrive.

Power Consumption

Ultrasonic and Hall Effect sensors are the most power-efficient we've listed, here, while LIDAR sensors are the least.

MEMS

Okay, so temperature and distance are more or less self-explanatory. But what exactly is MEMS? Well, it's an acronym that stands for Micro-ElectroMechanical System. It covers tiny, fabricated machines, some of which are smaller than a micron wide. These machines form the basis of the accelerometers, gyroscopes and magnetometers which allow your phone to understand how many steps you've taken in a given day, for example.



How are they built?

The production of MEMS devices takes advantage of the same fabrication methods widely used in integrated circuit manufacture. As such, they can be made efficiently and cheaply. There are two approaches to the 'micromachining' through which MEMS devices are made: bulk and surface. The former is a subtractive method which involves progressively removing layers of material from a surface using chemical or laser etching. The latter, meanwhile, is additive – with tiny layers of material being built in layers, much like in a stereo-lithographic 3D printer.

Accelerometers

Accelerometers measures acceleration forces. These forces could be static, like the constant force of gravity, or dynamic, like those caused by the movement of the device.

There's more than one way to build an accelerometer, but each includes micro-mechanical structures which move under force and thereby cause a change in the device's electrical properties. Many come with a tiny piezoelectric crystal inside, which, when stressed by sudden movement, will produce a measurable voltage. Others come with miniature structures that collectively carry a capacitance, which changes as one of these structures moves relative to the other.

More important than what goes on inside the accelerometer is the output it produces. You'll find considerable variety, here; some accelerometers communicate digitally, while others put out a variable analog voltage. The Arduino can read both, but a digital accelerometer will free up computational resources, as it won't need regular polling via an analogRead() command.

Nowadays, most accelerometers measure forces along three axes, and can vary in sensitivity and maximum swings We used SparkFun's triple-axis accelerometer breakout board when we built our 'Arduino Pet' project. The pet will generate R2D2-style beeps and bloops whenever it changes feels some movement. All you need to do is drag it around on the end of a lead.

We also used an accelerometer in our animatronic tail project, specifically a combined triple-axis accelerometer and gyro, again from Sparkfun.

Speaking of gyros...

Gyro Sensors

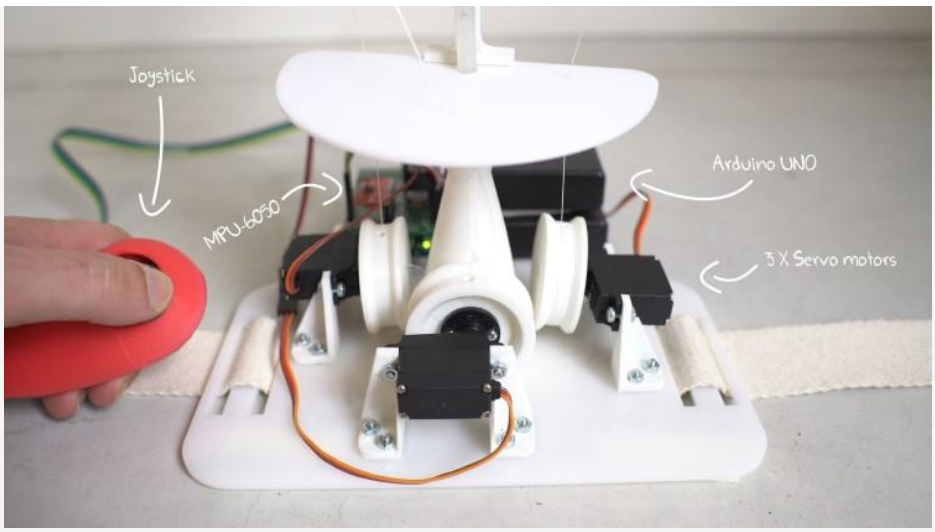
A gyro sensor carries out a similar function to the accelerometer, except that it's built to sense changes in angular velocity. You'll find them on the wheels of cars, inside video-game controllers, and in the shake-correction hardware in a high-quality camera. They tend to work using a series of arms which bend towards and away from one another as the apparatus rotates.

Gyro sensors can measure angular velocity, and this information can be used to calculate the angles themselves over a longer period of time. They can also be used to measure (and correct for) vibration. If you're building robots, or need to measure precise inputs from the physical world, then you're definitely going to be using a gyro sensor. Much like accelerometers, they come with a range of outputs and sensitivities, and so you'll want to pick the one that best matches your project.

Magnetometer

A magnetometer measures magnetic fields in three axes relative to the earth's magnetic field. They're used in digital compasses for example. While there are some other technologies available, the overwhelming majority of modern magnetometers work via the hall-effect principle we've outlined.

Although you can find each of the MEMS sensors as a separate sensor, it's common to find them in combinations of 2 or even 3. For example: the MPU6050 we used in our animatronic tail project incorporates both an accelerometer and a gyro.



How to choose the right MEMS sensor?

MEMS are unlike the other types of sensor we've mentioned here, since they're defined by how they're made rather than by the thing they measure. It's not possible to compare accelerometers, gyros and magnetometers directly, as each serves a distinct purpose. What we *can* compare are different types of accelerometers, gyros and magnetometers: for example, DC versus AC accelerometers. This goes a little beyond the scope of this article – but we may just return to the topic in a future entry!

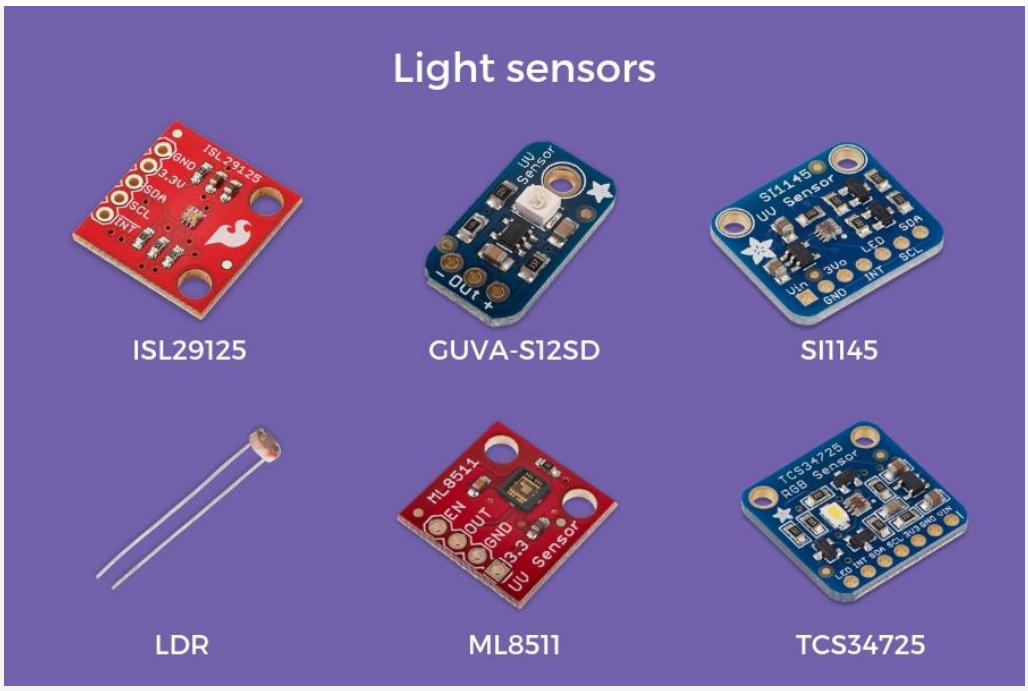
Color/Light Sensors

The spectrum is light is very wide, consisting mainly of Infrared, visible light and ultraviolet. It is measured in frequency or wavelengths.

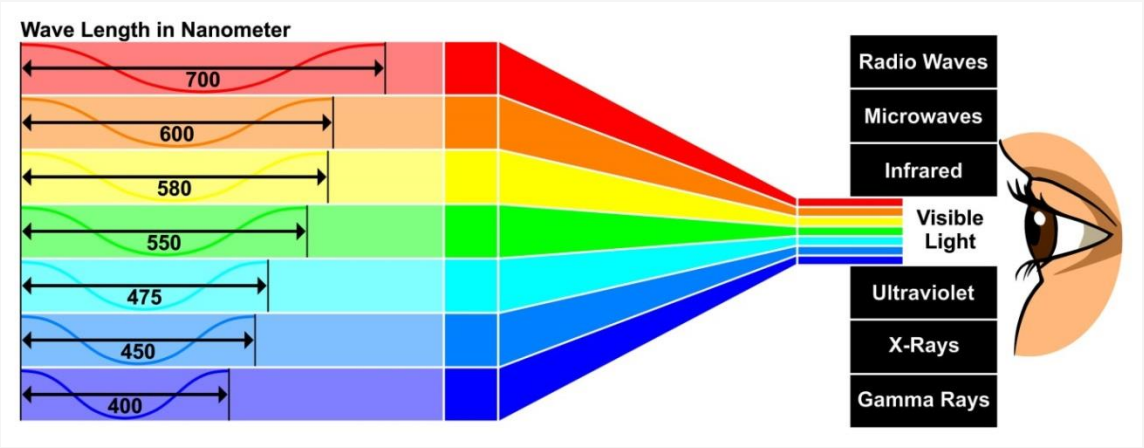
$$f = \frac{v}{\lambda}$$

f - frequency [Hz] v - speed of light [$\frac{meter}{second}$] λ – wavelength [$meter$]

To measure the different parts of the spectrum, you'll need to use different sensors that respond to the light frequency you want to measure.

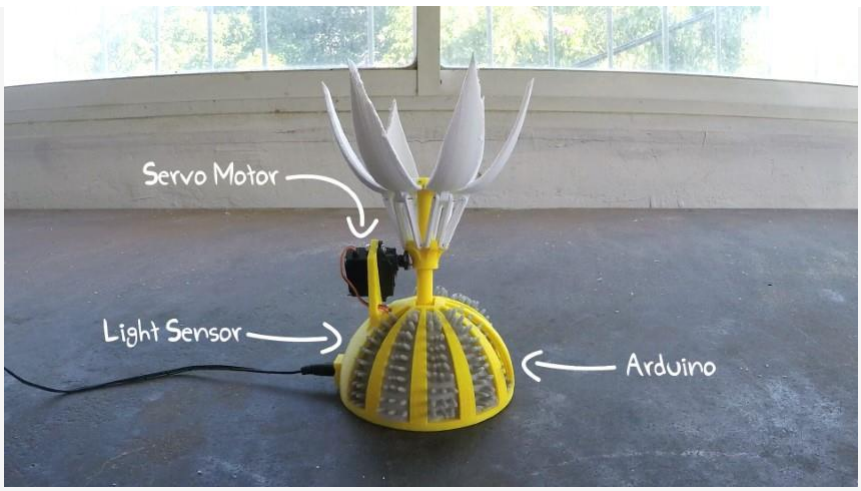


Every color in the visible light spectrum has a different frequency as you can see in the figure below:



LDR

The simplest form of light-sensor is the Light Dependent Resistor, or LDR (or photo-resistor). The LDR resistance drops when exposed to light, compared to the resistance value measured in the dark. In other words, the measurement sampled from the LDR is that of the resistance and not the light itself. We've used and LDR in our mechanical sunflower, whose 3D-printed petals unfold whenever the sun rises.



RGB Light Sensor

An RGB light sensor works by sending out pure white light and measuring the wavelength of the reflection. The response is sent through three separate filters so that the red, blue and green components are turned into a current, which in turn is transmitted to the Arduino via a digital serial communication.

Light sensors form the basis of the CCD sensor found in digital cameras, but they're also extremely useful in industrial sorting machines, where they're used to filter manufacturing defects out of production lines at lightning pace. But they're also great for domestic use. For example, we've put together an Arduino-based circuit for detecting the color of a coffee capsule and displaying the relevant information on an LCD screen. And once you've gotten it working, you'll be able to apply the same circuitry to a range of other solutions.

How to choose a light sensor

Spectrum (IR, specific color, ambient light)

The complex information put out by an RGB sensor allows you to sense for a specific combination of colors. By contrast, a photo-resistor will simply vary according to the overall incoming light.

Temperature Sensitivity

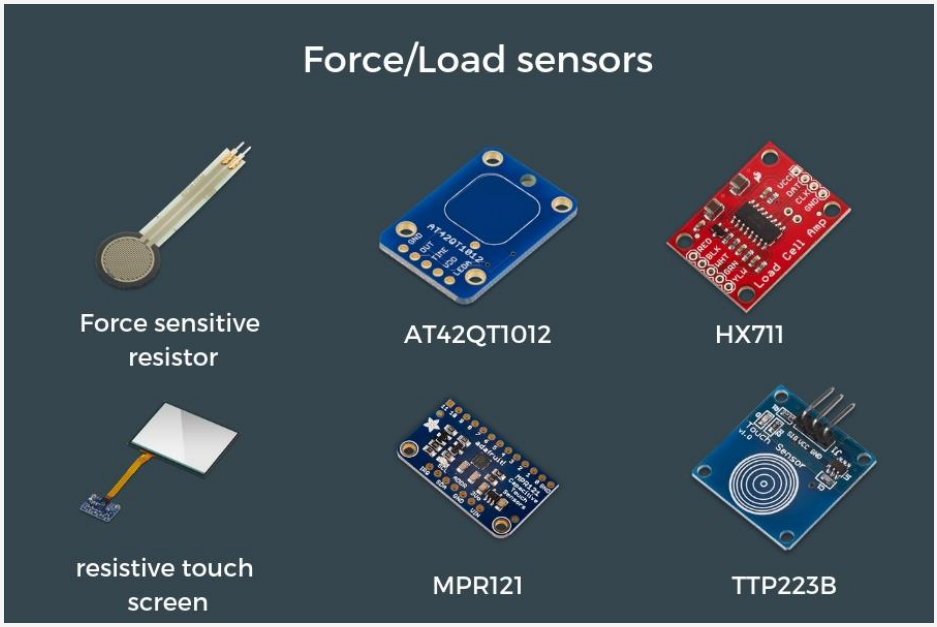
Given that LDRs will vary in resistance as the temperature rises and falls, they're a poor match for situations where the temperature is going to fluctuate.

Refresh rate

An LDR can respond to a small change in light in a few milliseconds, but if going from total light to total darkness, or vice versa, it can take a second or longer for the change to fully register. RGB sensors are comparatively zippy.

Touch/Force Sensors

Force and touch sensors convert physical force into electrical signals. Like the other sensors we've mentioned, these come in several different varieties, each of them suitable for a slightly different application. Force-sensitive sensors are found in a many devices such as mobile phones, digital scales and also in your car, as part of the airbag triggering system.



Touchscreens

The most obvious sort of force-sensitive sensor is the one most of us carry around in our pockets every day: the touchscreen on our mobile phones. Touchscreens come in two varieties: resistive and capacitive.

Resistive touchscreens are made from layers of conductive material, arranged to detect the presence and location of pressure at a given moment. They're able to respond to even very small amounts of pressure, from just about any object, but they aren't quite as fast to update as the capacitive touchscreen you have on your phone.

Capacitive touchscreens are built differently, and rely on the conductive properties of the human finger to work. A sheet of transparent conductor is placed behind a sheet of insulator, like glass. When a finger is pushed against the latter, it'll cause a change in the electrostatic field of the former. This information can be used (through software) to quickly deduce where the finger is. Even if there are multiple contacts simultaneously, the computer will be able to work out where they are.

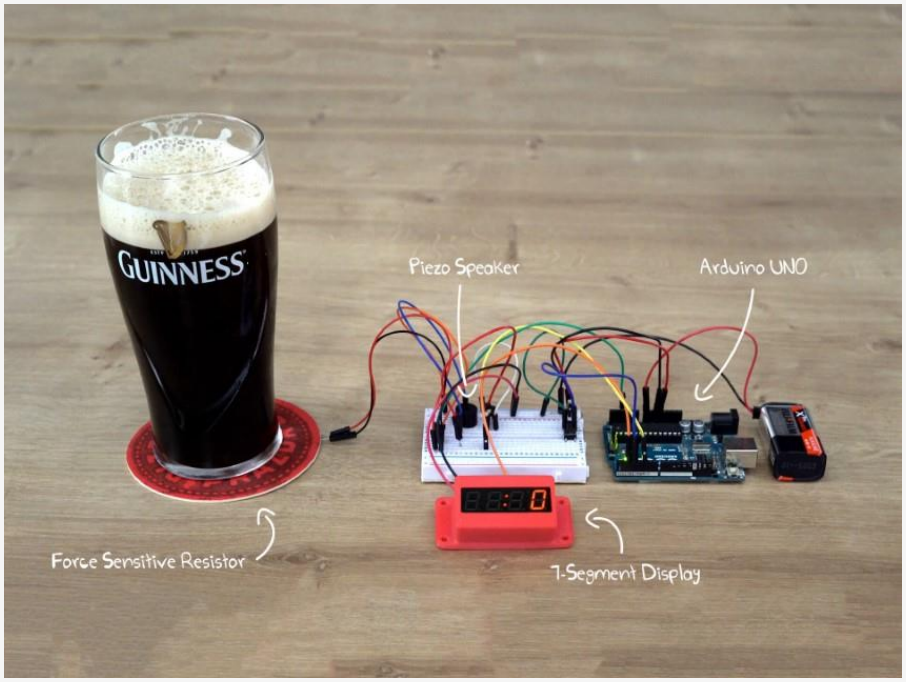
FSRs

A force-sensing resistor is a kind of mechanical pressure sensor built from a conductive print, consisting of a zig-zag of narrow traces, placed slightly below a conductive film. As pressure is applied, the two move closer to one another, and the current is able to flow more

easily through the entirety. You can even make your own FSRs using a piece of conductive sponge. Squash it, and the space within will shrink, allowing current to flow more easily.

FSRs are very cheap. You can usually pick one up for just a few dollars. However, they aren't very precise – and so they make poor matches for electronic scales and other such pieces of equipment.

But for certain applications, these extra features aren't necessary. We used an FSR in our St. Patrick's day project, the 'chug-meter'. It combines an FSR with a coaster to measure the weight of a pint of Guinness, starting a timer once the pint is off the coaster, and stopping it when the pint is put back.



Load Cell

Strain-gauge load cells work very differently than force-sensitive resistors, but they exploit the same basic principle. When a tightly-packed region of zig-zagging wiring is compressed, the resistance decreases. The 'strain gauge' itself is a zig-zag pattern printed onto a flexible substrate. When force is applied, it becomes more conductive. A typical strain-gauge load switch incorporates four of these devices, with an average being taken from each. Thus most load cells are accurate to within 0.1%, which compares very favorably with their FSR counterparts. There are other varieties of load cell available, including hydraulic and pneumatic ones, but thanks to their lower production costs, strain gauges represent a growing majority of load-cells available.

Load cells, of every sort, come with their downsides. They tend to be bulky and expensive, and they can be unreliable if overloaded, improperly mounted, or exposed to a corrosive environment. This is in marked contrast with FSRs, which are a little more robust.

We've used this component in our April-fool 'Evil fruit bowl' which squeals in agony (or plays any sound you decide to load up) every time a piece of fruit is removed from it. It's the accuracy of the load cell that allows for this to happen consistently – it's precise enough to be able to detect the difference when a single banana is removed from a fully-loaded fruit bowl.

How to choose a touch/force sensor

Accuracy

Load cells are the most accurate type of force sensor. while FSRs are the least.

Max/min force

A load cell can also tolerate far greater differences in pressure than any other sort of force sensor. The more expensive ones can precisely weight up to a hundred kilos or more.

Interface

A resistive touchscreen, with its digital connectivity, offers the most straightforward interface of the sensors we've listed.

Footprint

The footprint on a force-sensitive resistor can be made very small indeed - and they're also completely flat. By contrast, a strain-gauge load cell is extremely bulky and thus unsuitable for most home-electronics applications.

Conclusion

Hopefully by now you've got an idea of the sorts of sensors available and what they do. There's plenty more detail to get into, and there's plenty of information out there on each of the sensors we've mentioned. If you want to try to get to grips with any of them, then why not start by putting together some of the sample projects we've linked to?

Reading 3 "An introduction to Arduino pinout"

Zait, A, (2018) *'An introduction to Arduino pinout'*, Circuito.io blog, Roboplan Technologies Ltd, https://www.circuito.io/blog/arduino-uno-pinout/, viewed 19 December 2018

AN INTRODUCTION TO ARDUINO UNO PINOUT

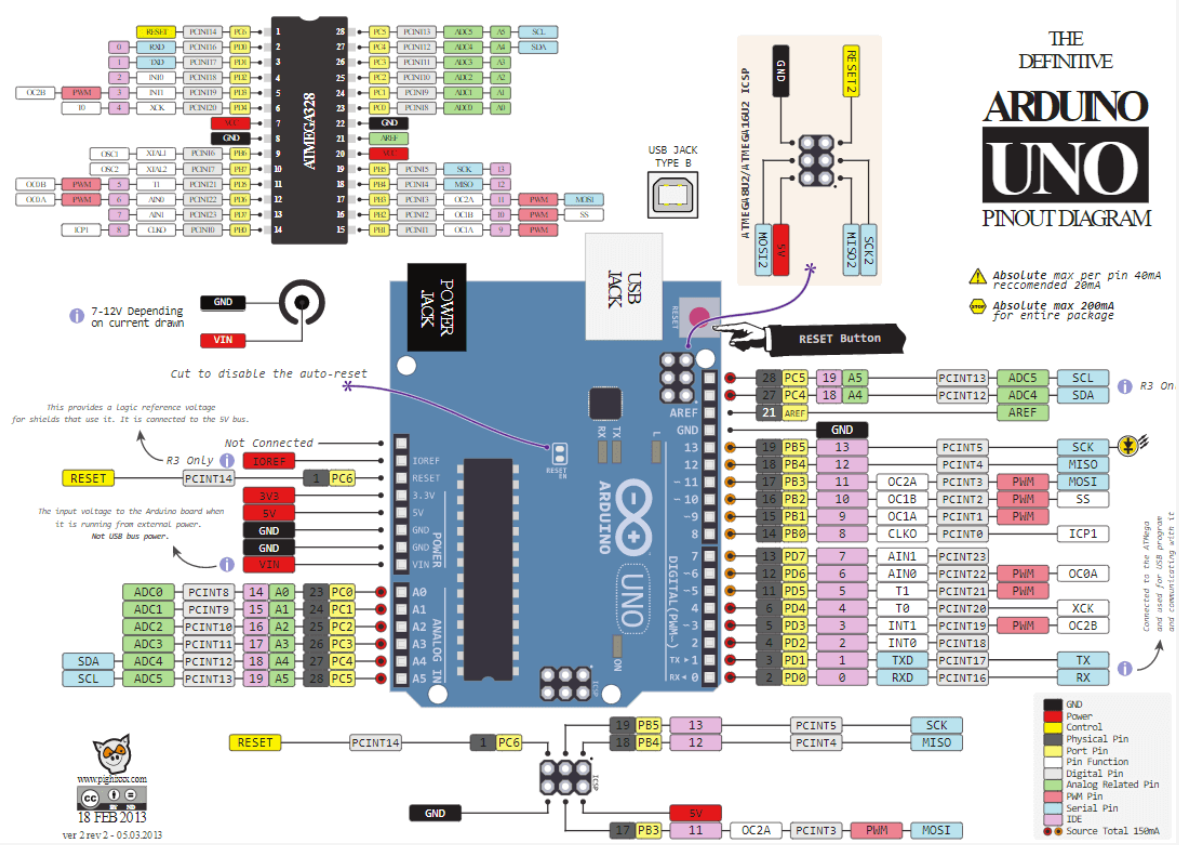
BLOG POST ANAT ZAIT APRIL 22, 2018

Arduino Uno Pinout Guide

In our last two posts, we focused on the software aspects of the Arduino. We saw that Arduino boards are programmed using a language derived from C and C++ in Arduino's Integrated Development Environment (IDE) and learned a few basic debugging methods. In this post, we'll be taking a closer look at the Arduino hardware, and more specifically, the Arduino Uno pinout. Arduino Uno is based on the ATmega328 by Atmel. The Arduino Uno pinout consists of 14 digital pins, 6 analog inputs, a power jack, USB connection and ICSP header. The versatility of the pinout provides many different options such as driving motors, LEDs, reading sensors and more. In this post, we'll go over the capabilities of the Arduino Uno pinout.

Start Your Arduino Circuit

Arduino Uno Pinout - Diagram

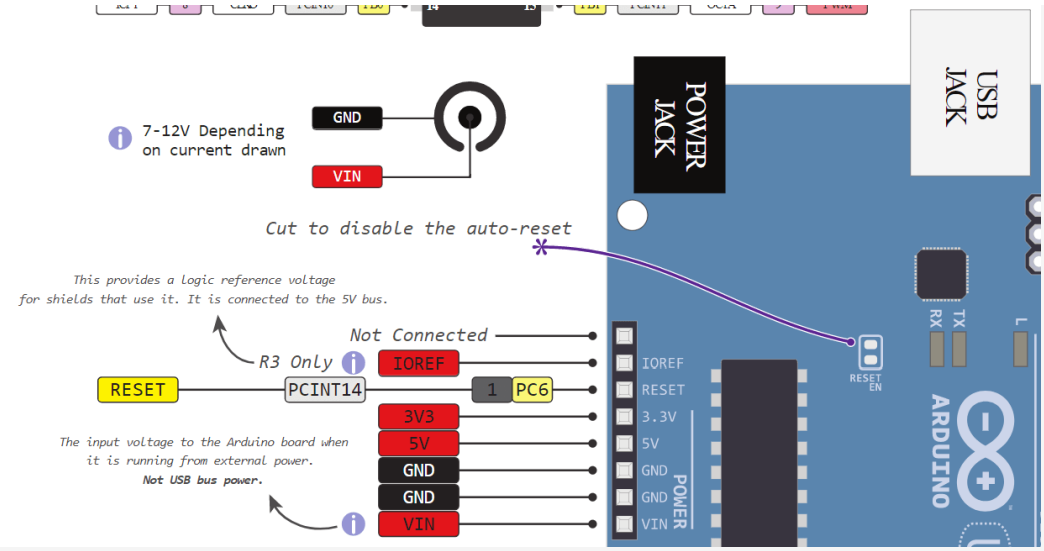


"Pinout of ARDUINO Board and ATMega328PU" by pighixxx is licensed under Creative Commons Attribution-Share Alike 4.0 International

Arduino Uno pinout - Power Supply

There are 3 ways to power the Arduino Uno:

- **Barrel Jack** - The Barrel jack, or DC Power Jack can be used to power your Arduino board. The barrel jack is usually connected to a wall adapter. The board can be powered by 5-20 volts but the manufacturer recommends to keep it between 7-12 volts. Above 12 volts, the regulators might overheat, and below 7 volts, might not suffice.
- **VIN Pin** - This pin is used to power the Arduino Uno board using an external power source. The voltage should be within the range mentioned above.
- **USB cable** - when connected to the computer, provides 5 volts at 500mA.



There is a polarity protection diode connecting between the positive of the barrel jack to the VIN pin, rated at 1 Ampere.

The power source you use determines the power you have available for your circuit. For instance, powering the circuit using the USB limits you to 500mA. Take into consideration that this is also used for powering the MCU, its peripherals, the on-board regulators, and the components connected to it. When powering your circuit through the barrel jack or VIN, the maximum capacity available is determined by the 5 and 3.3 volts regulators on-board the Arduino.

- **5v and 3v3**

They provide regulated 5 and 3.3v to power external components according to manufacturer specifications.

- **GND**

In the Arduino Uno pinout, you can find 5 GND pins, which are all interconnected.

The GND pins are used to close the electrical circuit and provide a common logic reference level throughout your circuit. Always make sure that all GNDs (of the Arduino, peripherals and components) are connected to one another and have a common ground.

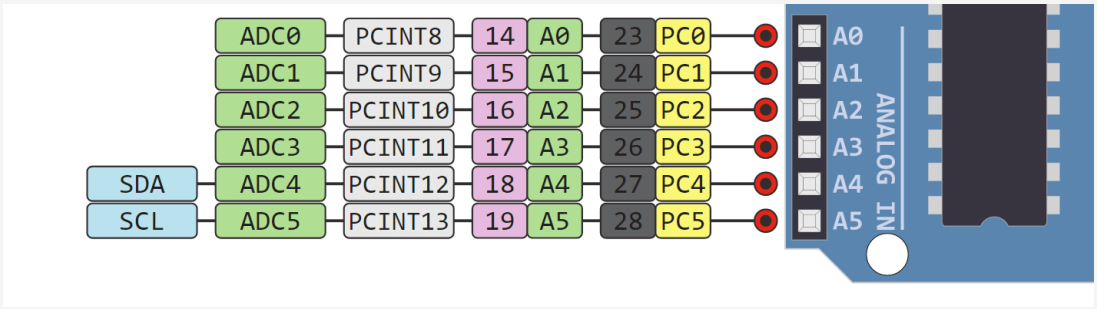
- **RESET** - resets the Arduino

- **IOREF** - This pin is the input/output reference. It provides the voltage reference with which the microcontroller operates.

Arduino Uno Pinout - Analog IN

The Arduino Uno has 6 **analog pins**, which utilize ADC (Analog to Digital converter).

These pins serve as analog inputs but can also function as digital inputs or digital outputs.



Analog to Digital Conversion

ADC stands for Analog to Digital Converter. ADC is an electronic circuit used to convert analog signals into digital signals. This digital representation of analog signals allows the processor (which is a digital device) to measure the analog signal and use it through its operation.

Arduino Pins A0-A5 are capable of reading analog voltages. On Arduino the ADC has 10-bit resolution, meaning it can represent analog voltage by 1,024 digital levels. The ADC converts voltage into bits which the microprocessor can understand.

One common example of an ADC is Voice over IP (VoIP). Every smartphone has a microphone that converts sound waves (voice) into analog voltage. This goes through the device's ADC, gets converted into digital data, which is transmitted to the receiving side over the internet.

Arduino Uno Pinout - Digital Pins

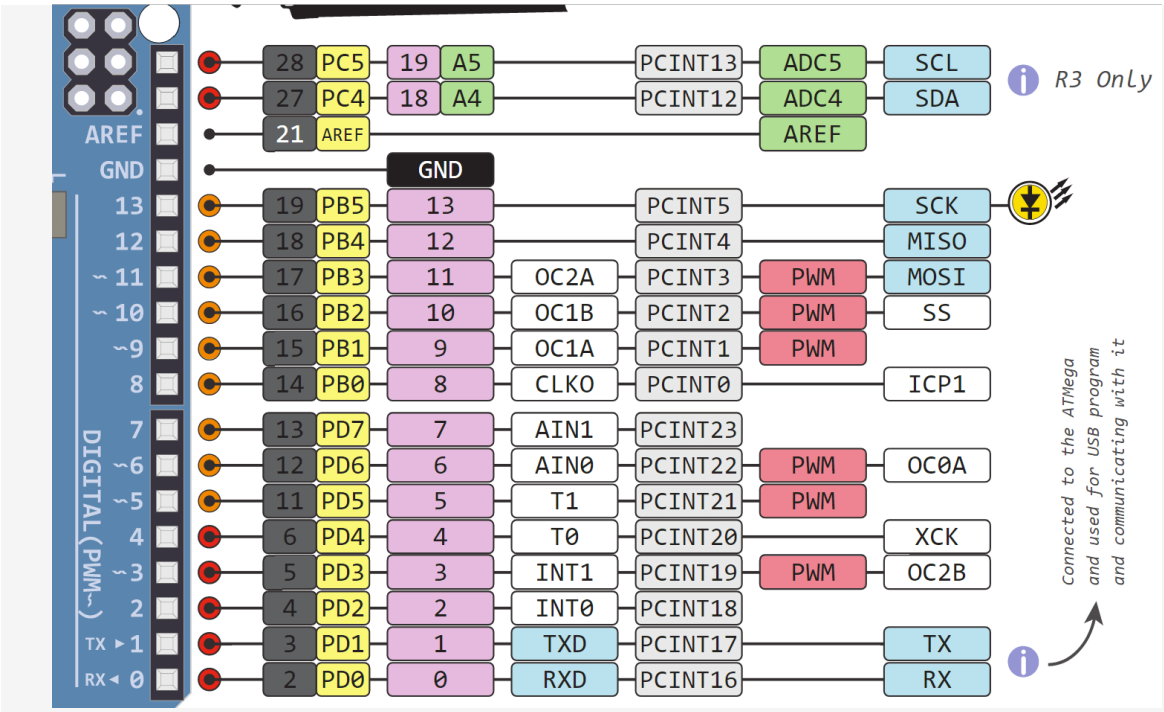
Pins 0-13 of the Arduino Uno serve as digital input/output pins.

Pin 13 of the Arduino Uno is connected to the built-in LED.

In the Arduino Uno - pins 3,5,6,9,10,11 have PWM capability.

It's important to note that:

- Each pin can provide/sink up to 40 mA max. But the recommended current is 20 mA.
- The absolute max current provided (or sank) from all pins together is 200mA



What does digital mean?

Digital is a way of representing voltage in 1 bit: either 0 or 1. Digital pins on the Arduino are pins designed to be configured as inputs or outputs according to the needs of the user. Digital pins are either on or off. When ON they are in a HIGH voltage state of 5V and when OFF they are in a LOW voltage state of 0V.

On the Arduino, When the digital pins are configured as **output**, they are set to 0 or 5 volts.

When the digital pins are configured as **input,** the voltage is supplied from an external device. This voltage can vary between 0-5 volts which is converted into digital representation (0 or 1). To determine this, there are 2 thresholds:

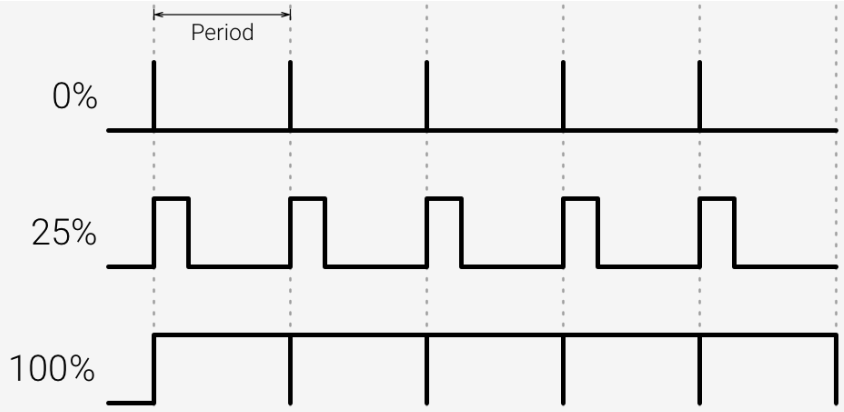
- Below 0.8v - considered as 0.
- Above 2v - considered as 1.

When connecting a component to a digital pin, make sure that the logic levels match. If the voltage is in between the thresholds, the returning value will be undefined.

What is PWM?

In general, Pulse Width Modulation (PWM) is a modulation technique used to encode a message into a pulsing signal. A PWM is comprised of two key components: **frequency** and **duty cycle**. The PWM frequency dictates how long it takes to complete a single cycle (period) and how quickly the signal fluctuates from high to low. The duty cycle determines how long a signal stays high out of the total period. Duty cycle is represented in percentage.

In Arduino, the PWM enabled pins produce a constant frequency of ~ 500Hz, while the duty cycle changes according to the parameters set by the user. See the following illustration:



PWM signals are used for speed control of DC motors, dimming LEDs and more.

Communication Protocols

Serial (TTL) - Digital pins 0 and 1 are the serial pins of the Arduino Uno.

They are used by the onboard USB module.

What is Serial Communication?

Serial communication is used to exchange data between the Arduino board and another serial device such as computers, displays, sensors and more. Each Arduino board has at least one serial port. Serial communication occurs on digital pins 0 (RX) and 1 (TX) as well as via USB. Arduino supports serial communication through digital pins with the SoftwareSerial Library as well. This allows the user to connect multiple serial-enabled devices and leave the main serial port available for the USB.

Software serial and hardware serial - Most microcontrollers have hardware designed to communicate with other serial devices. Software serial ports use a pin-change interrupt system to communicate. There is a built-in library for Software Serial communication. Software serial is used by the processor to simulate extra serial ports. The only drawback with software serial is that it requires more processing and cannot support the same high speeds as hardware serial.

SPI - SS/SCK/MISO/MOSI pins are the dedicated pins for SPI communication. They can be found on digital pins 10-13 of the Arduino Uno and on the ICSP headers.

What is SPI?

Serial Peripheral Interface (SPI) is a serial data protocol used by microcontrollers to communicate with one or more external devices in a bus like connection. The SPI can also be used to connect 2 microcontrollers. On the SPI bus, there is always one device that is denoted as a Master device and all the rest as Slaves. In most cases, the microcontroller is the Master device. The SS (Slave Select) pin determines which device the Master is currently communicating with.

SPI enabled devices always have the following pins:

- MISO (Master In Slave Out) - A line for sending data to the Master device
- MOSI (Master Out Slave In) - The Master line for sending data to peripheral devices
- SCK (Serial Clock) - A clock signal generated by the Master device to synchronize data transmission.

I2C - SCL/SDA pins are the dedicated pins for I2C communication. On the Arduino Uno they are found on Analog pins A4 and A5.

What is I2C?

I2C is a communication protocol commonly referred to as the “I2C bus”. The I2C protocol was designed to enable communication between components on a single circuit board. With I2C there are 2 wires referred to as SCL and SDA.

- SCL is the clock line which is designed to synchronize data transfers.
- SDA is the line used to transmit data.

Each device on the I2C bus has a unique address, up to 255 devices can be connected on the same bus.

Aref - Reference voltage for the analog inputs.

Interrupt - INT0 and INT1. Arduino Uno has two external interrupt pins.

External Interrupt - An external interrupt is a system interrupt that occurs when outside interference is present. Interference can come from the user or other hardware devices in the network. Common uses for these interrupts in Arduino are reading the frequency a square wave generated by encoders or waking up the processor upon an external event.

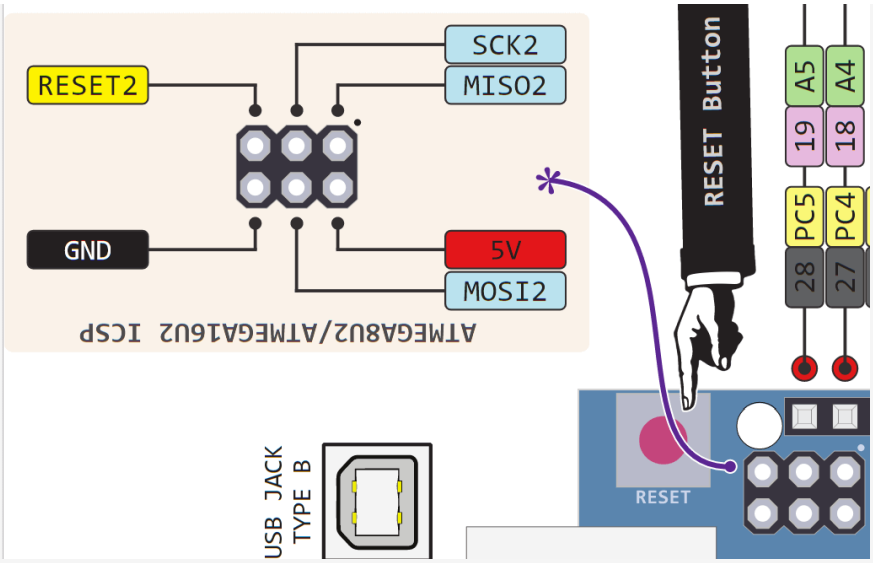
Arduino has two forms of interrupt:

- External
- Pin Change

There are two external interrupt pins on the ATmega168/328 called INT0 and INT1. both INT0 and INT1 are mapped to pins 2 and 3. In contrast, Pin Change interrupts can be activated on any of the pins.

Arduino Uno Pinout - ICSP Header

ICSP stands for In-Circuit Serial Programming. The name originated from In-System Programming headers (ISP). Manufacturers like Atmel who work with Arduino have developed their own in-circuit serial programming headers. These pins enable the user to program the Arduino boards’ firmware. There are six ICSP pins available on the Arduino board that can be hooked to a programmer device via a programming cable.



Know your Pinout

The Arduino Uno Microcontroller is one of the most versatile boards on the market today and that's why we decided to focus on it in this guide. This guide displays most of its capabilities, but there are also more advanced options which we did not go into in this post.

The important thing to know when you choose a board for your project is its capabilities and limitations. It's also important to understand the different communication protocols that the board uses. Of course, you don't need to remember all of this information, you can always go back to this post and read the relevant information for you.

Reading 4 "4 simple steps for debugging your Arduino project"

Zait, A, (2018) *'4 simple steps for debugging your Arduino project'*, Circuito.io blog, Roboplan Technologies Ltd, https://www.circuito.io/blog/arduino-debugging/, viewed 19 December 2018

4 SIMPLE STEPS FOR DEBUGGING YOUR ARDUINO PROJECT

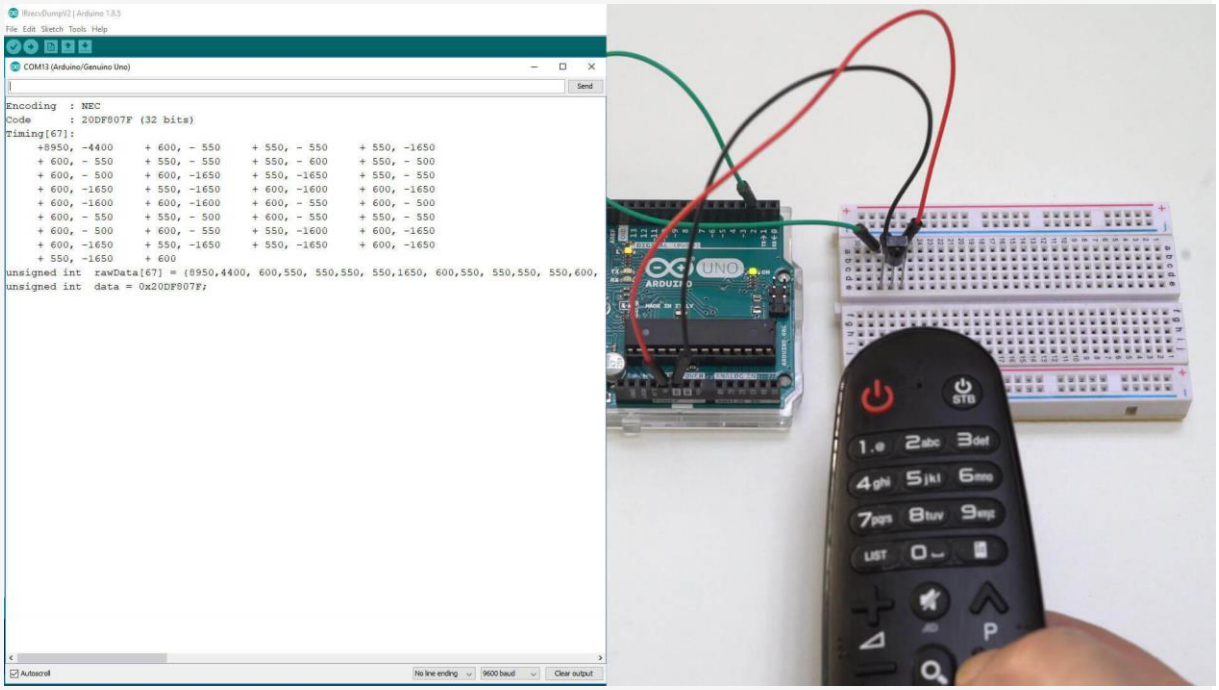
BLOG POST ANAT ZAIT APRIL 2, 2018

How to Debug Your Arduino Project

Anyone who's written code in the past can tell you that it usually doesn't work as initially expected. In fact, when it comes to writing code, debugging is an essential part of the creative process. However, Arduino debugging can be a little more complicated. One of the reasons for that is that Arduino doesn't come with a custom debugging tool.

Start Your Circuit >>

Most programs will use a debugger to help the programmer locate bugs and address them. Yet Arduino has no such debugging system. Arduino Debugging is one of the more challenging tasks of managing an Arduino project. Unlike most other IDEs, there is no official Arduino debugging feature onboard the Arduino IDE. Many users are surprised at the absence of a dedicated Arduino debugging tool.

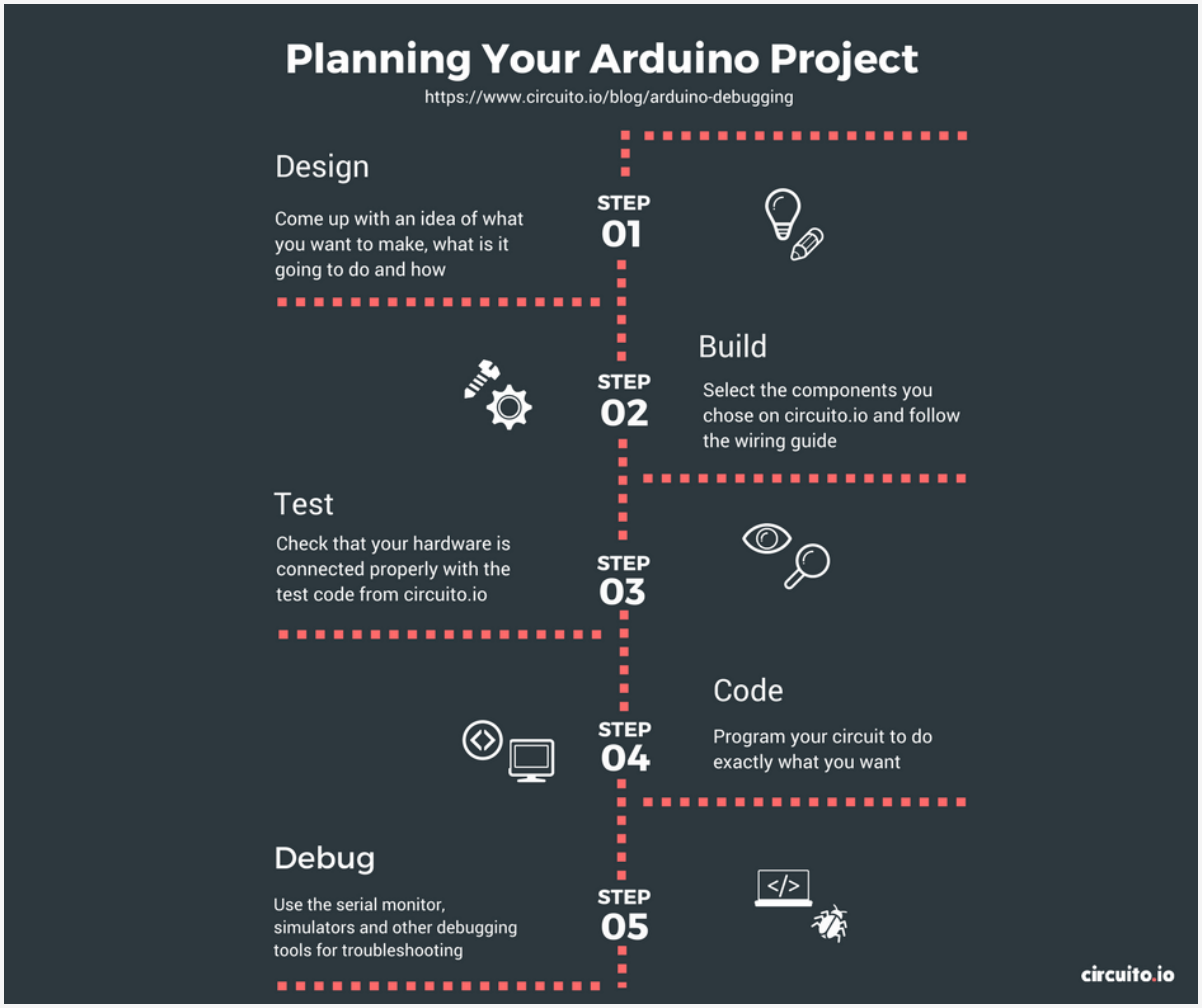


On Arduino, once you run your program you are unable to see what's happening inside your device and how your code is running. The only assistance you're given comes in the form of messages on your serial monitor or by LED display. Especially when other IDEs usually have debugging tools that utilize breakpoints, steps, call stack, watch, local/global variables and other components to double check through code.

However, debugging Arduino code isn't the same as debugging code on a PC. The two are very different in terms of how they are approached. The main reason for this is that Arduino code is usually used to control physical outputs or receiving physicals inputs to/from the real world and the debugging process has to take those into account.

In practice, this means that Arduino developers often have to explore alternative methods and tools to debug their code. In this post, we'll break down everything from Arduino code

debugging to using a simulator to debug Arduino. This will give you all the information you need to produce code that works.



Design

When it comes to creating an Arduino Project your project's overall design is very important. From the moment you start choosing components for your project, you need to have a clear idea of your end product in mind.

For example, if you want to design a sketch that runs on an Arduino board and lights up LEDs, you need to provide the foundations within the design phase of your project. The overall design of your sketches and your circuits will determine how your finished project looks.

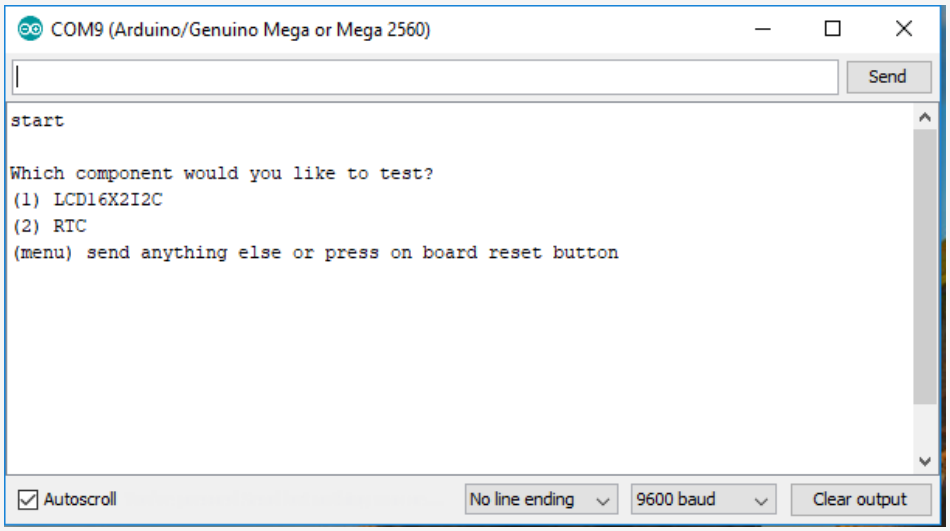
Build

In terms of building your project, this is where you actually give life to your circuit. The 'build process' is where the Arduino IDE compiles your C++ sketch into machine code which is then uploaded straight to your Arduino board.

Circuito.io **Code**

The code you get from circuito.io is actually a test code without any logic. This test code is used to check that all your components are wired properly and function correctly. This helps

to check that there are no faults with your hardware setup. circuito.io also provides you with code libraries for all the different components you chose. If your test code fails, then you know you need to start the debugging process.



Step 1: Check your hardware

As mentioned above, if your circuito.io code doesn't work, chances are you have a problem with your hardware. This means you need to begin Hardware debugging.

Here are some basic troubleshooting steps for hardware issues:

1. **Check wiring -** One of the first things you should do when you wire your circuit is to check that you have connected everything correctly. If you've put the jumper wires in the wrong pin of the Arduino, or the wrong placement on the breadboard, your program will not be able to interact with the component. Double-check all the components and wires to make sure everything is securely in place. If you have a complex circuit, sometimes it's just better to start from scratch, testing each component at a time to eliminate the problem.
2. **Check Soldering -** If some of your parts required soldering during setup, make sure that you soldered them properly. Poorly soldered parts are one of the most common causes of technical failure. If you can't tell, resolder it.
3. **Check Power Supply -** If there's still a problem with your circuit, you need to check the power supply. Using a multimeter, you can check 2 things:

Step 2: Code debugging

If your test code works, it's time to write your own code. When writing your own Arduino code, there are a number of basic rules and best practices you should follow. By following the instructions below, you'll have a much easier time if you need to debug it later on:

- Write code in small chunks and test each of them.
- Provide meaningful names for variables and functions
- Use functions
- Use constants rather than numbers
- Write comments to explain coding choices for future reference
- Make sure your code has a proper indentation and remains readable at all times. you can use Alt-t to auto-indent the whole sketch.

Compilation

If you get a compilation error when trying to compile or upload code to your board check for errors in syntax, typos and more. Using the correct syntax is vital for making sure your code compiles. When compilation fails, the IDE will present you with the errors on its bottom part. However, the error messages generated by the Arduino IDE are limited in their description and therefore not always very helpful.

If you understand what the error message means, try fixing the problem it indicates and compiles again. This is a convenient form of Arduino troubleshooting.

However, if you address an error listed on the IDE and the sketch still doesn't compile, it's a good idea to use Google to search for solutions to the problem.

Common syntax errors are:

- missing ; at the end of each line
- misplaced or missing parenthesis { }
- typo
- undefined variables and functions

Arduino has a large open source community with extensive Arduino troubleshooting guides, which will help you to identify problems with your code. Once you've solved the problem and your code works, click run.

Run code

If your code compiles and uploaded successfully to the board but doesn't run as expected you need to begin debugging.

Serial Monitor

As outlined above, if your code fails when you try to run your sketch, you need to start debugging your code. First, think and define which parameters to print and use the serial monitor to monitor them on screen(You can learn the basics of serial print here). The aim is to print an overview of the current state of the program. As such:

- Variables
- Inputs – sensor readings
- Prints that indicate the programs flow, like inside an 'if' statement to see whether the condition was met
- Outputs – e.g. PWM values before writing them to the pin
- Anything else you find important to print to screen

Check Your Code Manually

One good way to assess the quality of your code and check for errors is by going through your code manually. You can do this by:

1. Writing your code from scratch and breaking it down to see if you made any mistakes along the way (Most commonly, these will be syntax errors).
2. Revisiting your design and asking what your main logic is. Write it down in words and then see if your code statements answer this question. Sometimes when we go deep into programming we make the mistake of focusing on the small details and missing the overall direction of the project. In this state, it's easy to lose connection to the initial idea.
3. Go over your code and add comments (as much as needed) to explain to yourself what you are trying to do.

Step 3: Using external software Debugging tools

If you check your code manually and still can't find the problem, it's time to use an advanced debugging tool. Whereas many IDE's have their own onboard debugging tool, Arduino doesn't. However, there are a number of external tools you can use to make sure your code is running correctly. Here are some of the best tools to consider if you need advanced debugging and simulation options:

Visual Micro

Visual micro is a plugin available via Microsoft Visual Studio that is used to create cross-platform programs on Arduino. Any code created in Visual Micro that adheres to Arduino will be accepted. Visual Micro is great for collaborative teams debugging Arduino because it enables shared code and library editing. Code can be created across different platforms and combined with the program code throughout the build process. Visual Micro also offers GDB debugging and Serial, Bluetooth and Wifi debugging.

Atmel studio

Atmel Studio IDE is a free software that offers competitive debugging facilities to help resolve code errors. Through Atmel Studio, a project can be developed, compiled and uploaded to the relevant microprocessor. Like Arduino IDE, Atmel Studio's IDE uses the same code, meaning that you don't need to learn a new programming language in order to use it. Atmel is very versatile for the open source community and supports .ino sketches and C++ source codes. Atmel Studio also provides very good debugging capabilities using DebugWire or JTAG.

DebugWIRE

DebugWire is a protocol of Atmel to debug many ATTiny (e.g. ATTiny 13, 85) and ATmegas (e.g. ATmega48/88/168/328) without JTAG, only via the Reset pin. The DebugWire protocol is not documented by Atmel but some guys reverse engineered big parts of the protocol, and were able to build some simple debuggers. By using debugWIRE one has full read and write access to all memory and full control over the execution flow. It supports single-step, run-to-cursor, step-out, and software break instructions.

Step 4: Using Arduino simulators and emulators

More tools you can use for monitoring and debugging are Arduino emulators and simulators. Arduino simulators have made it easier than ever before for experts and hobbyists alike to program and test their ideas until they run efficiently. Hardware simulation is a complex process and while in the industry there are amazing tools for hardware debugging these tools are quite limited for makers and hobbyists.

However, simulators and emulators still have a place among the Arduino user's debugging toolkit. Arduino simulators support line-to-line debugging which allows the user to look through their code and establish where they went wrong. Here are just a few advantages of using simulators:

Debugging

As mentioned above, simulators are great for debugging Arduino, both in terms of syntax and functional errors. What makes simulators suitable for debugging is that you can write code and create electronic circuits to test the integrity of your code. Some simulators will offer you a limited library of hardware to test whereas others will allow you to develop complex virtual environments. Today, you can even use simulators to render your project in 3D.

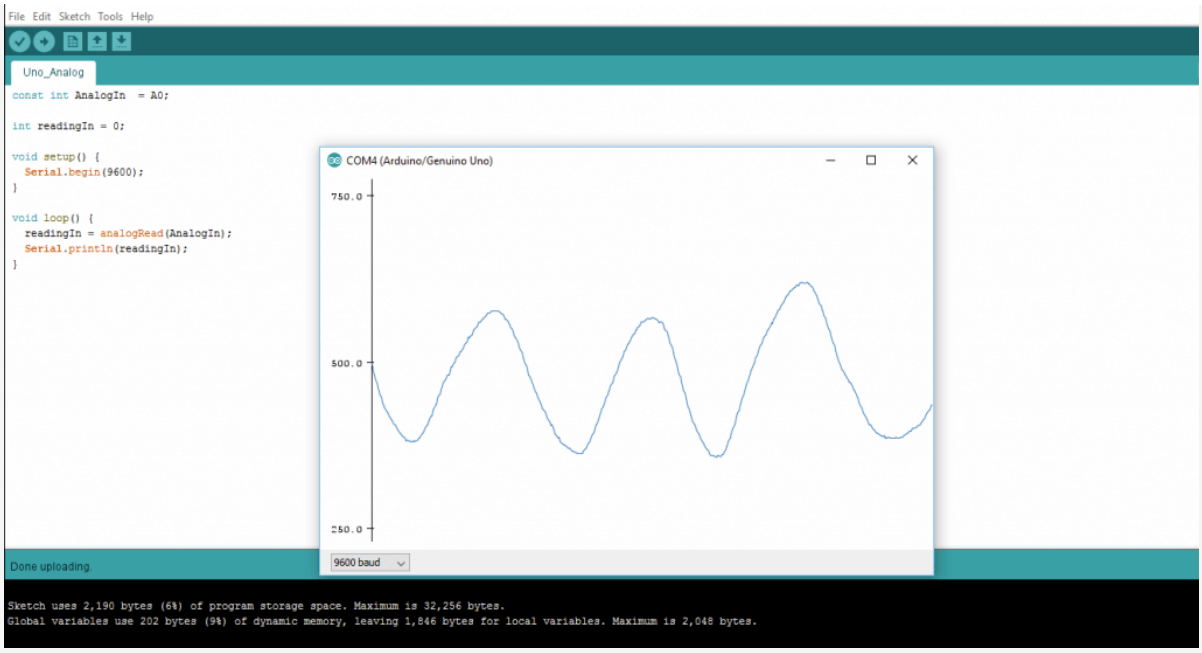
You can see what the program does

One of the biggest advantages provided by simulators is transparency. When running a simulation you can see exactly how your code works and identify ways it can be improved. This allows you to run new code without worrying about damaging your board or equipment.

Compared to IDE and hardware setups, simulators allow users to correct functional programming errors. Without a simulator, you can only address non-functional, technical mistakes in your code such as syntax errors. This makes simulators ideal for newer users who will need to undergo a lot of trial and error before they reach the finished product. Testing code via a simulator ensures that hardware stays operational, saving time and money.

Plotting and logging

Simulators not only allow you to test how your code runs, but allow you to log and plot the data generated as well. You can take note of your programming data and plot it on an external program like Excel. Logging data on your program and your additional devices helps to develop insights that can help refine your coding.



Experimentation

In terms of experimentation, simulators and emulators are hard to beat as well. Without a simulator, the user is confined to creating code based on their theoretical knowledge and has limited opportunities to try out new code and new components (especially when mistakes result in damaged hardware). With a simulator, you can test the code in a virtual environment and try out new ideas without worrying about the end result.

Testing new components

Likewise, simulators offer you a way to test out components before you pull the trigger and make a purchase. This way you can see if a part will be of use to your project, and practice integrating it into your overall environment. This ensures that you don't waste money on buying parts that are of little to no use.

Create blueprints for electronic circuits

When constructing an electronic circuit, you can use a simulator to design, build and preview schematics. This cuts out much of the legwork of needing to manually draw up blueprints for electronic circuits saving you time. In addition, it also ensures that the end circuit design is adequately optimized.

Popular Simulator

1. **Electronics Lab on Tinkercad (formerly circuits.io)**

The Circuits.io platform makes it easy for users to simulate real-world electronics through a circuit simulator. This online simulator allows the user to drag in an Arduino board and start programming. It also offers the user a circuit diagram maker, multimeter measurement tool and oscilloscope. This Arduino simulator allows you to build your design from scratch whilst enabling you to take precise measurements of the power supply throughout your circuit.

2. **Virtual Breadboard**

Virtual Breadboard has established a name for itself as one of the most powerful and advanced Arduino simulators available today. Virtual Breadboard has grown within the electronic circuit industry and today can simulate Arduino devices, Netduino and PIC

microcontrollers. On start up the user has access to a complete virtual development environment where they can program directly to an Arduino board. It's also worth noting that it can act as an AVR emulator as well.

Debugging Arduino Doesn't Need to be Difficult

When you start using Arduino, debugging can seem like a complicated process, but if you break it down into steps, you will probably be able to overcome the problems and learn a lot along the way.

As you saw in this guide, there are a variety of external tools to help you but the best way to debug is by trial and error, identifying faults along the way and fixing them as you go.

Another method that will help you cut down on the number of errors you encounter is to be strict with your syntax. This will ensure that when the time comes to debug your code you won't have to sift through syntax errors which could have been eliminated during your write up.

Compared to an IDE with an onboard debugger, debugging an Arduino can be quite inconvenient. If you test, compile and run your code before putting it through a third party debugger, you'll be well on track to producing quality code. Likewise, if you get lost along the way there's a vibrant open-source community on hand to help.